CONDITION NUMBER ANALYSES OF LINE/SURFACE AND SURFACE/SURFACE INTERSECTION ALGORITHMS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Gun Srijuntongsiri

January 2008

CONDITION NUMBER ANALYSES OF LINE/SURFACE AND SURFACE/SURFACE INTERSECTION ALGORITHMS

Gun Srijuntongsiri, Ph.D.

Cornell University 2008

Intersection problems have many applications in computational geometry and geometric modeling and design. This dissertation addresses two specific intersection problems: finding all intersections between a line and a parametric surface and between two parametric surfaces. New algorithms based on Newton's method and subdivision are proposed to solve these problems. Our algorithms also use a test based on the Kantorovich theorem to prevent the divergence or slow convergence issues normally associated with using unsuitable starting points for Newton's method. The algorithm for line/surface problem in particular can operate on polynomials represented in any basis that satisfies a few conditions. The power basis, Bernstein, and first-kind Chebyshev bases are among those compatible with the algorithm. The novelty of our algorithms is the analyses showing that their running time is bounded only in terms of the condition number of the problem instance and, in the line/surface case, the constant depending on the polynomial basis. This constant measures the tightness of the bounding polytope as compared to the bounded subsurface, which translates to the efficiency of the algorithm when the basis is used. The constant is different for each basis as each one lends itself to computation of different bounding polytope. We derive this constant for the three mentioned commonly used bases.

BIOGRAPHICAL SKETCH

Gun Srijuntongsiri was born in Bangkok but moved to Nakhon Sawan, which is a province in the middle part of Thailand, when he was three. He attended the elementary and middle schools in his hometown. In 1994, he attended Triam Udom Suksa school, one of the prestigious high schools in Thailand at the time. He won the King's scholarship in 1996, which financially supported him during his undergraduate education abroad.

Having decided to continue his study in the United States, he studied at Phillips Exeter Academy in New Hampshire, USA, for a year. He had his undergraduate education majoring in computer science from Cornell University. In 2002, he began his Ph.D. study in the field of computer science at Cornell University. His research interest is in numerical analysis, semidefinite programming, and computational geometry.

To my family.

ACKNOWLEDGEMENTS

I would like to thank my Ph.D. advisor, Dr. Stephen Vavasis, for his guidance during my entire study and for being a great mentor. He has offered me many great insights and ideas, most of which are present in this dissertation, especially when I was stuck and could not proceed with the research by myself. I also would like to thank my other special committee members Dr. Michael J. Todd and Dr. Stephen R. Marschner for their time and valuable suggestions to my research and dissertation, the anonymous referee of one of my manuscripts for suggesting that using the Chebyshev basis with my line/surface intersection algorithm might make it more efficient, and NSF and NSERC (Canada) for financially supporting the research in this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Intersection problems have many applications in areas such as geometric modeling and design, computational geometry, robotics, collision avoidance, manufacturing simulation, scientific visualization, and computer graphics. This dissertation addresses two important types of intersection problems: finding all of the intersections between a line and a parametric surface and between two parametric surfaces. The parametric method of surface representation is a very convenient way of approximating and designing curved surfaces, and computation using parametric representation is often much more efficient than other types of surface representations. Moreover, it is also more widely used in practice than other kinds of surface representations.

Typically, intersection problems involving nonlinear geometric elements such as surfaces reduce to solving systems of nonlinear equations. There are a number of algorithms proposed in the literature to solve these two intersection problems. We review the important ones in the following sections.

1.1 Review of line/surface intersection algorithms in the literature

The very first algorithms for solving the line/surface intersection problem are the subdivision methods introduced by Whitted [35, 27]. In these methods, a simple shape such as a rectangular box or a sphere is used to bound the surface. The

bounding volume is then tested to see whether it intersects the line. If it does, the surface patch is subdivided, and the bounding volumes are found for each subpatch. The process repeats until no bounding volumes intersect the line or the volumes are smaller than a specified size where the intersections between such volumes and the line are taken as the intersections between the surface and the line. Subdivision methods are robust and simple, but normally not efficient when high accuracy of the computed solutions is required. They also cannot indicate if there is more than one zero inside the remaining subdomains.

Regardless, a variation of subdivision methods known as Bézier clipping by Nishita et al. should be noted for its efficient subdivision [22]. For a non-rational Bézier surface, Bézier clipping uses the intersection between the convex hull of the orthographic projection of the surface along the line and a parameter axis to determine the regions which do not contain any intersections before subdividing the remaining region. Sherbrooke and Patrikalakis generalize Bézier clipping to a zero-finding algorithm for an n-dimensional nonlinear polynomial system called *Projected Polyhedron* (PP) algorithm [29].

On the numerical side, Kajiya [16] proposes a method for intersecting a line with a bicubic surface based on algebraic geometry without subdivisions. His method is robust and simple. However, it is too costly to extend to higher degree polynomials. Jónsson and Vavasis [15] introduce an algorithm for solving systems of two polynomials in two variables using Macaulay resultant matrices. They also analyze the accuracy of computed zeros in term of the conditioning of the problem.

Another approach is to combine a subdivision method with a Newton-type method, using the latter to find the solutions of the resulting system of equations after subpatches pass some criteria. The advantages of Newton's method are its

quadratic convergence and simplicity in implementation, but it requires a good initial approximation to converge and does not guarantee that all zeros have been found. To remedy these shortcomings, Toth [33] uses a result from interval analysis to determine the "safe regions", where a Newton-like method starting from any point in them is guaranteed to converge. He tests each patch to see if it is a safe region and if its axis-aligned bounding box intersects the line. If neither is true, the patch is subdivided recursively. Lischinski and Gonczarowski [20] propose an improvement to Toth's algorithm specific to scene-rendering in computer graphics by utilizing ray and surface coherences.

In contrast, other researchers develop methods to estimate good initial points for Newton's method rather than test for convergence of each choice of initial points. These methods use tighter bounding volumes and subdivide the surface adaptively until subpatches are *flat* enough, that is, until they are close enough to the bounding volumes. Then the intersection between the bounding volume and the line is chosen as the initial point for Newton's method. Examples of methods in this category are [3, 10, 25, 31, 37]. There is also the ray-tracing algorithm by Wang et al. that combines Newton's method and Bézier clipping together [34].

1.2 Review of surface/surface intersection algorithms in the literature

Unlike line/surface, curve/curve, or curve/surface intersection problems, where the solutions usually are a (possibly zero) number of points, the solution of a surface/surface intersection problem typically contains multiple connected components. For this reason, different techniques are required to solve surface/surface

intersection problems. There are three main techniques for finding the intersections between two parametric surfaces: lattice, subdivision, and marching methods. Lattice methods treat one surface as a collection of isoparametric curves and reduce the problem to curve-surface intersections. The individual intersection points are then connected to form the intersection curves of the original surface-surface intersection [26]. A disadvantage of these methods is that their efficiency and accuracy depend heavily on the chosen grid resolution. By choosing the resolution too low, certain features of the intersections, e.g. small loops, may be missed. Choosing the resolution too high, on the other hand, leads to greater computation time in exchange for minimal increase in accuracy.

For subdivision methods, the basic idea is to subdivide the surfaces into smaller surfaces until they are either small or flat enough for their intersections to be approximated by the intersections between two simpler shapes such as planes [14, 23]. The individual solutions are then connected to form the complete solutions. The subdivisions are not necessarily uniform; an adaptive subdivision algorithm would subdivide certain parts of the surfaces with complicated geometry into smaller pieces than the parts with relatively simple geometry. Subdivision methods are robust, but if used by themselves for high-precision evaluation, they tend to be slow as large number of subdivisions are needed.

Marching methods generate sequences of points on an intersection curve by stepping from a given point on the curve in a direction according to the local differential geometry [1, 2, 36]. These methods are very efficient but they need to find a point on each of the intersection curve to use as a starting point for the tracing of the curve step. Open intersection curves always start and end on a surface boundary. Their starting points can therefore be located by solving a curve-

surface intersection between one surface and a border of another surface. On the other hand, the starting points of closed intersection curves, which are called loops, are not as simple to locate. One of the techniques to detect loops is by finding collinear normal points between the two surfaces, which provides points inside all loops and singular points—isolated intersection points [28]. Another well-known loop detection technique is based on oriented distance function, defined as the distance between a point on one surface and the point on the other surface closest to it as measured along a certain orientation, as the surface normals at the critical points of the gradient of the distance function are collinear [4]. The Poincaré index theorem can also be used to conclusively detect these critical points [19, 21]. However, both collinear normal points and oriented distance function methods are only applicable to surfaces whose normals do not vary too much.

Some methods are hybrid as they combine ideas from different techniques, usually from subdivision and marching ones. Grandine and Klein [13] identify the structure of the intersection curves by topology resolution before using a numerical tracing method to find the actual curves. Koparkar [18] subdivides the surfaces until a Newton-like method is guaranteed to find the intersection curves contained in the subsurfaces. The convergence test is based on the contraction mapping theorem and evaluating ranges of functions. Koparkar's algorithm is quite similar to Toth's algorithm for line/surface intersection problems and can be considered a generalization of the latter to surface/surface problems. The convergence tests used by the two algorithms are different but both are based on the same contraction mapping theorem.

1.3 Overview of our contributions

We propose new algorithms for solving line/surface and surface/surface intersection problems. Our algorithms are similar to Toth's and Koparkar's in that they subdivide the parametric domains of the problem until the subdomains pass certain tests before starting to locate the solutions by an iterative method. Another similarity is in the use of a bounding polytope of a subsurface to quickly determine if certain subdomains do not contain any solutions. Our convergence tests are based on Kantorovich's theorem, which tells us if Newton's method converges quadratically for the initial point in question in addition to whether it converges at all. For this reason, we can choose to hold off Newton's method until quadratic convergence is assured.

The main feature of our algorithms is that there are upper bounds on the number of subdivisions performed during the course of the algorithm that depend only on the condition number of the problem instance and, in the line/surface case, a constant depending on the polynomial basis. For example, having a solution located exactly on the border of a subdomain does not adversely affect its efficiency. To the best of our knowledge, there are no previous algorithms in this class whose running times have been bounded in terms of the condition numbers of the underlying problem instance, and we are not sure whether such an analysis is possible for previous algorithms. In particular, we do not know of any surface/surface intersection algorithm in the literature that has any *a priori* bound on the running time.

The notion of bounding the running time of an iterative method in terms of the condition number of the instance is an old one, with the most notable example being the condition-number bound of conjugate gradient (see Chapter 10 of [12]).

This approach has also been used in interior-point methods for linear programming [11] and Krylov-space eigenvalue computation [32].

Our algorithm for line/surface problem can operate on parametric surfaces represented in any basis satisfying a few conditions. The commonly used power, Bernstein, and first-kind Chebyshev bases are among those compatible with the algorithm. The choice of basis affects the type of bounding polygons used by the algorithm to determine if certain subpatches do not contain any zeros. The effectiveness of the convergence test is also affected by the choice of basis. The implication is that the efficiency of the algorithm depends on the basis and the type of bounding polygons suitable for that basis. For this reason, we introduce a constant to measure the tightness of a bounding polygon, which is the same constant that the running time of the algorithm depends on. We investigate the power, Bernstein, and first-kind Chebyshev bases in detail and derive their constants in Chapter 3.

Chapter 2

A Condition Number Analysis of a Line/Surface Intersection Algorithm

We begin with the simpler problem of the two, the line/surface intersection. In this chapter, we define the condition number of the line/surface intersection problem, propose a new algorithm to solve it, and analyze the running time of the algorithm in terms of its condition number. The contents of this chapter also appear in [30][1]. This new algorithm makes use of the result of Kantorovich's theorem, which is stated below.

2.1 The theorem of Kantorovich

Denote the closed ball centered at x with radius $r > 0$ by

$$\bar{B}(x, r) = \{y \in \mathbb{R}^n : \|y - x\| \leq r\},$$

and denote the interior of $\bar{B}(x, r)$ as $B(x, r)$. Kantorovich's theorem in affine invariant form is

Theorem 2.1.1 (Kantorovich, affine invariant form [5, 17]). *Let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ be differentiable in the open convex set D. Assume that for some point $x^0 \in D$, the Jacobian $f'(x^0)$ is invertible with*

$$\left\| f'(x^0)^{-1} f(x^0) \right\| \leq \eta.$$

Let there be a Lipschitz constant $\omega > 0$ for $f'(x^0)^{-1} f'$ such that

$$\left\| f'(x^0)^{-1} (f'(x) - f'(y)) \right\| \leq \omega \cdot \|x - y\| \text{ for all } x, y \in D.$$

[1]

If $h = \eta\omega \leq 1/2$ *and* $\bar{B}(x^0, \rho_-) \subseteq D$*, where*

$$\rho_- = \frac{1 - \sqrt{1-2h}}{\omega},$$

then f *has a zero* x^* *in* $\bar{B}(x^0, \rho_-)$*. Moreover, this zero is the unique zero of* f *in* $(\bar{B}(x^0, \rho_-) \cup B(x^0, \rho_+)) \cap D$ *where*

$$\rho_+ = \frac{1 + \sqrt{1-2h}}{\omega}$$

and the Newton iterates x^k *with*

$$x^{k+1} = x^k - f'(x^k)^{-1} f(x^k)$$

are well-defined, remain in $\bar{B}(x^0, \rho_-)$*, and converge to* x^**. In addition,*

$$\left\|x^* - x^k\right\| \leq \frac{\eta}{h}\left(\frac{(1-\sqrt{1-2h})^{2^k}}{2^k}\right), k = 0, 1, 2, \ldots \tag{2.1}$$

We call x^0 a *fast starting point* if the sequence of Newton iterates starting from it converges to a solution x^* and (2.1) is satisfied with $h \leq 1/4$, which implies quadratic convergence of the iterates starting from x^0. Kantorovich's theorem also holds for complex functions [9].

2.2 Formulation and representation of the line-surface intersection problem

Let $\phi_0, \ldots, \phi_n$ denote a basis for the set of univariate polynomials of degree at most n. For example, the power basis is defined by $\phi_i(t) = t^i$. Other choices of basis are discussed below.

Let S be a two-dimensional surface embedded in $\mathbb{R}^3$ parametrized by

$$\bar{f}(u, v) \;\; = \;\; \sum_{i=0}^{m} \sum_{j=0}^{n} \bar{c}_{ij} \phi_i(u) \phi_j(v), \quad 0 \leq u, v \leq 1,$$

where $\bar{c}_{ij} \in \mathbb{R}^3$ $(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$ denote the coefficients. Define a line

$$L \;=\; \{p + dt : \;\; t \in \mathbb{R}\},$$

where $p, d \in \mathbb{R}^3, d \neq 0$. The line-surface intersection problem is to find all of the intersections between S and L, which are the solutions of the polynomial system

$$\bar{f}(u, v) - (p + dt) = 0. \tag{2.2}$$

The system (2.2) can be reduced to a system of two equations and two unknowns. To show this, we first break (2.2) into its component parts

$$\begin{aligned} \bar{f}_1(u, v) - p_1 - td_1 &= 0, \\ \bar{f}_2(u, v) - p_2 - td_2 &= 0, \\ \bar{f}_3(u, v) - p_3 - td_3 &= 0. \end{aligned}$$

Here, the subscript i denotes the ith coordinate of the point in three-dimensional space. Assuming $|d_1| \geq \max\{|d_2|, |d_3|\}$, we have the equivalent system

$$\begin{aligned} d_1(\bar{f}_2(u, v) - p_2) - d_2(\bar{f}_1(u, v) - p_1) = 0, \\ d_1(\bar{f}_3(u, v) - p_3) - d_3(\bar{f}_1(u, v) - p_1) = 0, \end{aligned} \tag{2.3}$$

which can be rewritten with the same basis $\phi_i(u)\phi_j(v)$ (see item 4 on the list of basis properties below) as

$$f(u, v) \equiv \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij}\phi_i(u)\phi_j(v) = 0. \tag{2.4}$$

The system (2.4) is the one our algorithm operates on.

Since the parametric domain of the surface under consideration is square, our algorithm uses the infinity norm for all of its norm computation. Therefore, for the rest of this dissertation, the notation $\|\cdot\|$ is used to refer specifically to the infinity norm.

Our algorithm works with any polynomial basis $\phi_i(u)\phi_j(v)$ provided that the following properties hold:

1. There is a natural interval $[l, h]$ that is the domain for the polynomial. In the case of Bernstein polynomials, this is $[0, 1]$, and in the case of power and Chebyshev polynomials, this is $[-1, 1]$.

2. It is possible to compute a bounding polytope P of $S = \{f(u, v) : l \leq u, v \leq h\}$, where $f(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij}\phi_i(u)\phi_j(v)$ and $c_{ij} \in \mathbb{R}^d$ for any $d \geq 1$, that satisfies the following properties:

   (a) Determining whether $0 \in P$ can be done efficiently (ideally in $O(mn)$ operations).

   (b) The polytope P is affinely invariant. In other words, the bounding polytope of $\{Af(u, v) + b : l \leq u, v \leq h\}$ is $\{Ax + b : x \in P\}$ for any nonsingular matrix $A \in \mathbb{R}^{d\times d}$ and any vector $b \in \mathbb{R}^d$.

   (c) For any $y \in P$,
   $$\|y\| \leq \theta \max_{l\leq u,v\leq h} \|f(u, v)\|, \tag{2.5}$$
   where θ is a function of m and n.

   (d) If $d = 1$, then the endpoints of P can be computed efficiently (ideally in $O(mn)$ time).

3. It is possible to reparametrize with $[l, h]^2$ the surface $S_1 = \{f(x) : x \in \bar{B}(x^0, r)\}$, where $x^0 \in \mathbb{R}^2$ and $r \in \mathbb{R} > 0$. In other words, it is possible (and efficient) to compute the polynomial $\hat{f}$ represented in the same basis such that $S_1 = \{\hat{f}(\hat{x}) : \hat{x} \in [l, h]^2\}$.

4. Constant polynomials are easy to represent.

5. Derivatives of polynomials are easy to determine in the same basis (preferably in $O(mn)$ operations).

We are generally interested in the case where $d = 2$. In this case, we call P a bounding polygon. Recall that P is a bounding polygon of S if and only if $x \in S$ implies $x \in P$.

As shown later in Section 2.6, the efficiency of our algorithm depends on θ. Hence, the choice of the basis affects the algorithm's performance as each basis allows different ways to compute bounding polytopes.

2.3 The Kantorovich-Test Subdivision algorithm for Line/Surface intersections

Before we detail our algorithm, we define notation and crucial quantities that are used by the algorithm and its analysis. Denote $x = (u, v)$ as a point in two-dimensional parametric space and $f(x) = f(u, v)$ as the value of f at x.

For a given zero x^* of polynomial f, let $\omega_*(x^*)$ and $\rho_*(x^*)$ be quantities satisfying the conditions that, first, $\omega_*(x^*)$ is the smallest Lipschitz constant for $f'(x^*)^{-1}f'$, i.e.,

$$\left\| f'(x^*)^{-1} \left(f'(x) - f'(y) \right) \right\| \leq \omega_*(x^*) \cdot \|x - y\| \text{ for all } x, y \in B(x^*, \rho_*(x^*)) \quad (2.6)$$

and, second,

$$\rho_*(x^*) = \frac{2}{\omega_*(x^*)}. \quad (2.7)$$

Since $\omega_*(x^*)$ is nondecreasing as $\rho_*(x^*)$ increases in (2.6) but $\rho_*(x^*)$ is decreasing as $\omega_*(x^*)$ increases in (2.7), there exists a unique pair $(\omega_*(x^*), \rho_*(x^*))$ satisfying

the above conditions, and this pair can be obtained by binary search. When more than one function is being discussed, we use $\omega_*^f(x^*)$ to denote $\omega_*(x^*)$ of the function f. Approximation to these two quantities, $\omega_*(x^*)$ and $\rho_*(x^*)$, are computed and made use of by the algorithm.

For clarity, we simply abbreviate $\omega_*(x^*)$ and $\rho_*(x^*)$ as ω_* and ρ_*, respectively, throughout the rest of this chapter when it is clear from the context to which x^* the quantities belong.

Straightforward application of the affine invariant form of Kantorovich's theorem with $x^0 = x^*$ and $D = B(x^*, \rho_*(x^*))$ yields the result that x^* is the unique zero of f in $B(x^*, \rho_*(x^*))$. In fact, the above definitions of $\omega_*(x^*)$ and $\rho_*(x^*)$ are chosen such that the ball that is guaranteed by Kantorovich's theorem to contain no other zeros than x^* is the largest possible.

Define

$$\gamma(\theta) = 1/\left(4\sqrt{\theta(4\theta+1)} - 8\theta\right),$$

where θ is as in (2.5). Observe that $1 \leq \gamma(\theta) \leq \left(\sqrt{5}+2\right)/4 \approx 1.0590$ since $\gamma(\theta)$ is a decreasing function for positive θ and $\theta \geq 1$ by the definition of a bounding polygon. Another quantity of interest is $\omega_{D'}$, which is defined as the smallest nonnegative constant ω satisfying

$$\begin{aligned} \left\|f'(x^*)^{-1}\left(f'(y) - f'(z)\right)\right\| \quad &\leq \quad \omega \cdot \|y - z\|, \quad y, z \in D', x^* \in [0,1]^2 \\ &\qquad\qquad \text{satisfying } f(x^*) = 0, \end{aligned} \tag{2.8}$$

where

$$D' = \left[-\gamma(\theta), 1 + \gamma(\theta)\right]^2. \tag{2.9}$$

The motivation of this definition of D' is that it contains all domains whose Lipschitz constants may be needed during the course of the algorithm. Denote ω_f as

the maximum of $\omega_{D'}$ and all $\omega_*(x^*)$

$$\omega_f = \max\{\omega_{D'}, \max_{x^* \in \mathbb{C}^2 : f(x^*)=0} \omega_*(x^*)\}.$$

Finally, define the condition number of f to be

$$\mathrm{cond}(f) = \max\{\omega_f, \max_{x^* \in \mathbb{C}^2 : f(x^*)=0, y \in [0,1]^2} \left\| f'(x^*)^{-1} f'(y) \right\|\}. \tag{2.10}$$

Note that in (2.8), x^* is restricted to zeros in $[0,1]^2$ whereas in (2.10), x^* ranges over all complex zeros of f. We defer the discussion of why (2.10) is a reasonable condition number until after the description of our algorithm.

We define the *Kantorovich test* on a region $X = \bar{B}(x^0, r)$ as the application of Kantorovich's Theorem on the point x^0 using $\bar{B}(x^0, 2\gamma(\theta)r)$ as the domain D in the statement of the theorem and $\|f'(x^0)^{-1} f(x^0)\|$ as η. For ω, we instead use $\hat{\omega} \geq \omega$, where $\hat{\omega}$ is defined by (2.12) below. The region X passes the Kantorovich test if $\eta\hat{\omega} \leq 1/4$ and $\bar{B}(x^0, \rho_-) \subseteq D'$, which implies that x^0 is a fast starting point.

The other test our algorithm uses is the exclusion test. For a given region X, let $\hat{f}_X$ be the polynomial in the basis $\phi_i(u)\phi_j(v)$ that reparametrizes with $[l,h]^2$ the surface defined by f over X. The region X passes the *exclusion test* if the bounding polygon of $\{\hat{f}_X(u,v) : l \leq u, v \leq h\}$ excludes the origin. Note that the bounding polygon used in this test must satisfy item 2 of the basis properties listed in Section 2.2.

We now proceed to describe our algorithm, the *Kantorovich-Test Subdivision algorithm for Line/Surface intersections* or KTS-LS in short. The algorithm maintains a queue Q of unexamined domain and a set S of safe regions throughout its computation.

Algorithm KTS-LS:

- Let Q be a queue with $[0,1]^2$ as its only entry. Set $S = \emptyset$.
- Repeat until $Q = \emptyset$
  1. Let X be the patch at the front of Q. Remove X from Q.
  2. If $X \not\subseteq X_S$ for all $X_S \in S$,
     - Perform the exclusion test on $X = \bar{B}(x^0, r)$
     - If X fails the exclusion test,
       (a) Perform the Kantorovich test on X
       (b) If X passes the Kantorovich test,
           i. Perform Newton's method starting from x^0 to find a zero x^*.
           ii. If $x^* \notin X_S$ for any $X_S \in S$ (i.e., x^* has not been found previously),
               * Compute $\rho_*(x^*)$ and its associated $\omega_*(x^*)$ by binary search.
               * Set $S = S \cup \{B(x^*, \rho_*(x^*))\}$.
       (c) Subdivide X along both u and v-axes into four equal subregions. Add these subregions to the end of Q.

A few remarks are needed regarding the description of the KTS-LS algorithm.

- The subdivision in step 2.c is performed regardless of the result of the Kantorovich test. In general, passing the Kantorovich test does not imply that there is only one zero in X.
- The check that the zero found by Newton's method is not a duplicate (step 2.b.ii) is necessary since the Kantorovich test may detect a zero outside X.

- If the Kantorovich test is not applicable for a certain patch due to the Jacobian of the midpoint being singular, the patch is treated as if it fails the Kantorovich test.

One property of KTS-LS is that it is affine invariant. In other words, left-multiplying f with a 2-by-2 matrix A prior to executing KTS-LS does not change its behavior. This is the main reason we define the condition number to be affine invariant. Define $g \equiv Af$. To see that our condition number is affine invariant, note that $g'(x)^{-1}g'(y) = [Af'(x)]^{-1}Af'(y) = f'(x)^{-1}A^{-1}Af'(y) = f'(x)^{-1}f'(y)$ for any $x, y \in \mathbb{R}^n$. Therefore, $\text{cond}(g) = \text{cond}(f)$. In contrast, simpler condition numbers such as Lipschitz constants for f' are not affine invariant and hence are not chosen for our analysis.

Since Toth's algorithm is the most similar one to KTS-LS, it is worthwhile to discuss the main differences between the two and the implications these differences make. First, Toth's uses the Krawczyk-Moore test and another unnamed test, both based on interval analysis, as the convergence test. These two tests guarantee linear convergence for the *simple* Newton iteration—a variation of Newton's method where the Jacobian of the initial point is used in place of the Jacobian of the current point in every iteration. With our Kantorovich test, KTS-LS starts Newton's method only when quadratic convergence is assured.

Another main difference is in the choice of domains for the convergence test. Toth's uses the subpatch X itself as the domain for the test. This choice may exhibit undesirable behavior when a zero lies on the border of a subpatch, which is not necessarily on or near the border of the original domain $[0, 1]^2$. For example, consider the function $f(u, v) = (u^2 - .25, v - .8)^T$ whose zeros are $(.5, .8)$ and $(-.5, .8)$. The patch $\{(u, v) : .5 \leq u \leq .5 + \epsilon, a \leq v \leq b\}$ does not pass either

of Toth's convergence tests for any $\epsilon > 0$ and any $a \leq .8 \leq b$ although the patch $\{(u, v) : .45 \leq u \leq .8, 0 \leq v \leq 1\}$, a large patch whose borders do not coincide with any zeros, does pass the Krawczyk-Moore test. This results in excessive subdivisions by Toth's algorithm. KTS-LS uses $\bar{B}(x^0, 2\gamma(\theta)r)$ as the domain for X to avoid this problem. Theorem 2.6.1 below shows that the Kantorovich test does not have trouble detecting the zeros located on the border of the subpatch.

2.4 Implementation details when using power, Bernstein, or Chebyshev bases

This section covers the implementation details of KTS-LS when the polynomial system is in the power, Bernstein, or Chebyshev bases. The power basis for polynomials of degree n is $\phi_k(t) = t^k$ $(0 \leq k \leq n)$. The Bernstein basis is $\phi_k(t) = Z_{k,n}(t) = \begin{pmatrix} n \\ k \end{pmatrix} (1-t)^{n-k} t^k$ $(0 \leq k \leq n)$. The Chebyshev basis is $\phi_k(t) = T_k(t)$ $(0 \leq k \leq n)$, where $T_k(t)$ is the Chebyshev polynomial of the first kind generated by the recurrence relation

$$\begin{aligned} T_0(t) &= 1, \\ T_1(t) &= t, \\ T_{k+1}(t) &= 2tT_k(t) - T_{k-1}(t) \text{ for } k \geq 1. \end{aligned} \tag{2.11}$$

2.4.1 Bounding polygons

We begin with the choices of l and h and the definitions of bounding polygons of the surface $S = \{f(u, v) : l \leq u, v \leq h\}$, where $f(u, v)$ is represented by one

Table 2.1: The value of θ's of the power, the Bernstein, and the Chebyshev bases and their corresponding bounding polygons.

Basis	θ
Bernstein	$\left(\sum_{i=0}^{m}\prod_{i'\neq i}\frac{\max\{\lvert m-i'\rvert,\lvert i'\rvert\}}{\lvert i-i'\rvert}\right)\left(\sum_{j=0}^{n}\prod_{j'\neq j}\frac{\max\{\lvert n-j'\rvert,\lvert j'\rvert\}}{\lvert j-j'\rvert}\right)$ $= O(m^{m+1}n^{n+1})$
Chebyshev	$2(m+1)(n+1)$
Power	$(m+1)(n+1)(3^{m+1}-1)(3^{n+1}-1)/2$

of the three bases, that satisfy the required properties detailed in Section 2.2. For Bernstein basis, the convex hull of the coefficients (control points), call it P_1, satisfies the requirements for $l = 0$ and $h = 1$. The convex hull P_1 can be described as

$$P_1 = \left\{ \sum_{i,j} c_{ij} s_{ij} : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\}.$$

For power and Chebyshev bases, the bounding polygon

$$P_2 = \left\{ c_{00} + \sum_{i+j>0} c_{ij} s_{ij} : -1 \leq s_{ij} \leq 1 \right\}$$

satisfies the requirements for $l = -1$ and $h = 1$. Note that P_2 is a bounding polygon of S in the Chebyshev case since $|T_k(t)| \leq 1$ for any $k \geq 0$ and any $t \in [-1, 1]$. Determining whether $0 \in P_2$ is done by solving a small linear programming problem. The value of θ for each case is summarized in Table 2.1. Refer to Chapter 3 for the derivation of θ for each of the three bases as well as the proofs that P_1 and P_2 satisfy all of the basis properties listed in Section 2.2.

2.4.2 Computation of a Lipschitz constant

Another step of KTS-LS that needs further elaboration is the computation of a Lipschitz constant in the Kantorovich test. The Lipschitz constant for $f'(x^0)^{-1}f' \equiv$

g is obtained from an upper bound on the derivative of g

$$g'(x) = \left(\frac{\partial^2 \left(f'(x^0)^{-1} f \right)_i (x)}{\partial x_j \partial x_k} \right)$$

for all $x \in X$. Let $\hat{g} \equiv \hat{g}_X$ be the polynomial in the same basis as f that reparametrizes with $[l,h]^2$ the surface defined by g over X. We have that

$$\begin{aligned} \max_{x \in X} \|g'(x)\| &= \max_{x \in [l,h]^2} \|\hat{g}'(x)\| \\ &= \max_{x \in [l,h]^2} \max_{\|y\|=1} \|\hat{g}'(x) y\| \\ &\leq \max_{x \in [l,h]^2} \max_{i} \sum_{j=1}^{2} \sum_{k=1}^{2} |\hat{g}'_{ijk}(x)| \\ &\leq 4 \max_{i,j,k} \max_{x \in [l,h]^2} |\hat{g}'_{ijk}(x)|. \end{aligned}$$

Note that each entry of $\hat{g}'$ can be written as a polynomial in the same basis as f (refer to property 5 of the basis). For this reason, an upper bound of $\max_{x \in [l,h]^2} |\hat{g}'_{ijk}(x)|$ can be computed as follows: Let P_{ijk} be the bounding polytope (bounding interval in this case) of $\{\hat{g}'_{ijk}(x) : x \in [l,h]^2\}$ computed in the same way as described in Section 2.4.1. The maximum absolute value of the endpoints of P_{ijk} (refer to property 2d of the basis) is an upper bound of $\max_{x \in [l,h]^2} |\hat{g}'_{ijk}(x)|$. Let $\hat{\omega}$ denote the Lipschitz constant computed in this manner, that is,

$$\hat{\omega} \equiv 4 \max_{i,j,k} \max_{x \in [l,h]^2} |\hat{g}'_{ijk}(x)|, \tag{2.12}$$

where $\max_{x \in [l,h]^2} |\hat{g}'_{ijk}(x)|$ is computed from the endpoints of its bounding interval.

2.5 Significance of our condition number

We now discuss the significance of (2.10) to the conditioning of the problem. In particular, we attempt to justify that the efficiency of any algorithm in the same

class as KTS-LS is dependent on (2.10). This class of algorithms being considered includes any algorithm that (i) isolates unique zeros with subdivision before finding them and (ii) will not discard a patch until the convex hull of its function values (which is clearly a subset of any possible bounding convex polygon) excludes the origin.

2.5.1 Condition number and the Kantorovich test

This section discusses the relationship between ω_f and the Kantorovich test. We show that, for any given zero x^* of an arbitrary f, there is a function $\bar{f}$ such that x^* is also a zero of $\bar{f}$, $f'(x^*) = \bar{f}'(x^*)$, $\omega_*^f(x^*) = \omega_*^{\bar{f}}(x^*)$, and $\bar{f}$ has another zero y^* with $\|y^* - x^*\| = \rho_*(x^*)$. For example, consider a zero $x^* = (.5, .5)$ of the function $f = (u^3 - 2.2u^2 + 1.55u - .35, v^2 - .7v + .1)^T$, of which $\rho_*(x^*) = .1$. A corresponding $\bar{f}$ with the above properties is $\bar{f} = (u^2 - .9u + .2, .3v - .15)^T$, which has zeros at $(.5, .5)$ and $(.4, .5)$. Since the Kantorovich test uses only the function value, its first derivative, and the Lipschitz constant, all of which are the same for f and $\bar{f}$ at x^*, the functions f and $\bar{f}$ are identical from the perspective of the Kantorovich test applied to x^*. Therefore, $\rho_*(x^*)$ is a reasonable number that quantifies the distance between x^* and its nearest other zero barring the usage of additional information. Consequently, ω_f, which is greater than or equal to $\omega_*(x^*) = 2/\rho_*(x^*)$ for all zeros x^* of f, describes the distance between the closest pairs of zeros of f. Therefore, the efficiency of any algorithm that isolates unique zeros is dependent on ω_f.

The function $\bar{f}$ with the above properties can be constructed as follows: Let

$x^* = (u^*, v^*)$, $f'(x^*) = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{pmatrix}$, and $\omega_*^f(x^*) = \omega$. If $|\alpha_4| \geq |\alpha_3|$,

$$\bar{f}(u,v) = \begin{pmatrix} \frac{\omega(\alpha_1\alpha_4 - \alpha_2\alpha_3)}{2\alpha_4}(u-u^*)^2 + \alpha_1(u-u^*) + \alpha_2(v-v^*) \\ \alpha_3(u-u^*) + \alpha_4(v-v^*) \end{pmatrix}. \tag{2.13}$$

Otherwise,

$$\bar{f}(u,v) = \begin{pmatrix} \alpha_1(u-u^*) + \frac{\omega(\alpha_1\alpha_4 - \alpha_2\alpha_3)}{2\alpha_3}(v-v^*)^2 + \alpha_2(v-v^*) \\ \alpha_3(u-u^*) + \alpha_4(v-v^*) \end{pmatrix}.$$

It is straightforward to verify that $\bar{f}(x^*) = 0$, $\bar{f}'(x^*) = f'(x^*)$, and $\omega_*^{\bar{f}}(x^*) = \omega$. We now show that $\|y^* - x^*\| = \rho_*(x^*)$ for the case where $|\alpha_4| \geq |\alpha_3|$. The other case can be verified in the same manner. Let $y^* = (u^* + \Delta u, v^* + \Delta v)$. Substituting y^* into (2.13) and setting it to zero yields

$$g(u^*+\Delta u, v^*+\Delta v) = \begin{pmatrix} \frac{\omega(\alpha_1\alpha_4 - \alpha_2\alpha_3)}{2\alpha_4}(\Delta u)^2 + \alpha_1\Delta u + \alpha_2\Delta v \\ \alpha_3\Delta u + \alpha_4\Delta v \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \tag{2.14}$$

Solving (2.14) yields

$$\begin{aligned} \Delta u &= -\frac{2}{\omega}, \\ \Delta v &= \frac{\alpha_3}{\alpha_4} \cdot \frac{2}{\omega}. \end{aligned}$$

Since $|\alpha_4| \geq |\alpha_3|$ and $\rho_*(x^*) = 2/\omega$, $\|y^* - x^*\| = \rho_*(x^*)$.

2.5.2 Condition number and the exclusion test

The other term in our condition number, $\max_{x^* \in \mathbb{C}^2 : f(x^*)=0, y \in [0,1]^2} \|f'(x^*)^{-1} f'(y)\|$, relates to the convex bounding polygon test—the test to determine whether the

convex bounding polygon of a subpatch contains the origin. We show that there exists a function f such that a patch $B(x^0, r)$ where x^0 is relatively close to a zero, fails the convex bounding polygon test if $r \geq O(1/\operatorname{cond}(f))$. Denote $x^0 = (u^0, v^0)$. Define the complex function $g(z) = (z - (u^0 - \epsilon - i\epsilon)) \cdot (z - (u^0 + \epsilon - i\epsilon))$, where $\epsilon \in \mathbb{R}$ and $0 < \epsilon < 1$. Consider the following function

$$\begin{aligned} f(u,v) &= \begin{pmatrix} \operatorname{Re}(g(u+iv)) \\ \operatorname{Im}(g(u+iv)) \end{pmatrix} \\ &= \begin{pmatrix} u^2 - v^2 - 2u^0u - 2\epsilon v - 2\epsilon^2 + (u^0)^2 \\ 2uv - 2u^0v + 2\epsilon u - 2\epsilon u^0 \end{pmatrix}, \end{aligned} \tag{2.15}$$

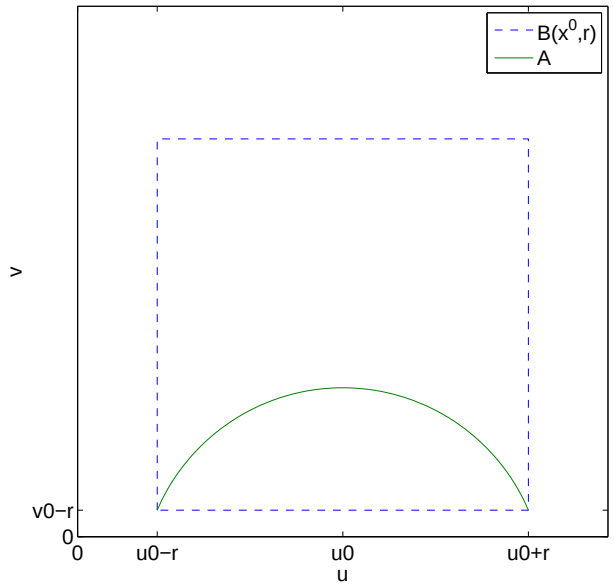
where $\operatorname{Re}(z)$ and $\operatorname{Im}(z)$ denote the real and imaginary parts of the complex number z, respectively. The four complex zeros of f are $(u^0 - \epsilon, -\epsilon)$, $(u^0 + \epsilon, -\epsilon)$, $(u^0, -\epsilon - i\epsilon)$, and $(u^0, -\epsilon + i\epsilon)$. Therefore,

$$\max_{x^* \in \mathbb{C}^2 : f(x^*) = 0, y \in [0,1]^2} \left\| f'(x^*)^{-1} f'(y) \right\| = O(1/\epsilon).$$

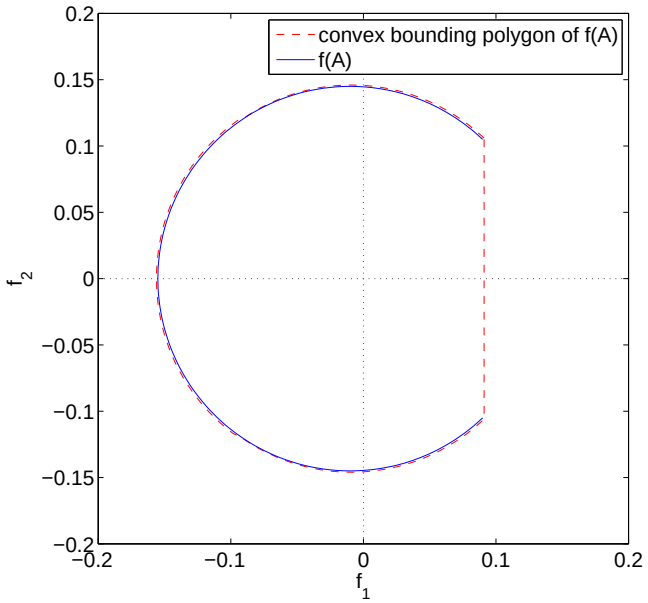
Moreover,

$$\omega_f = O(1/\epsilon).$$

We now show for the case that $v^0 = O(\epsilon)$ that $B(x^0, r)$ fails the convex bounding polygon test if $r \geq O(\epsilon)$. Let A be the circular arc centered at $(u^0, -\epsilon)$ that goes from $(u^0 + r, v^0 - r)$ to $(u^0 - r, v^0 - r)$ counterclockwise. Observe that f maps A to the circular arc centered at $(-\epsilon^2, 0)$ that goes from $(2(v^0 + \epsilon)r - 2\epsilon v^0 - (v^0)^2 - 2\epsilon^2, 2r(v^0 + \epsilon - r))$ to $(2(v^0 + \epsilon)r - 2\epsilon v^0 - (v^0)^2 - 2\epsilon^2, -2r(v^0 + \epsilon - r))$ counterclockwise (see Figure 2.1). Notice that $2r(v^0 + \epsilon - r) \geq 0$ because $B(x^0, r) \subseteq [0,1]^2$. Therefore, the convex bounding polygon of $f(A)$ contains the origin if $r > ((v^0)^2 + 2\epsilon v^0 + 2\epsilon^2)/(2(v^0 + \epsilon)) = O(\epsilon)$ (recall the assumption that $v^0 = O(\epsilon)$). Since $A \subset B(x^0, r)$, the convex bounding polygon of $f(B(x^0, r))$ also contains the origin and the convex bounding polygon test fails.



(a) The circular arc $A \subseteq \bar{B}(x^0, r)$.



(b) The range $f(A)$ and its convex bounding polygon

Figure 2.1: The circular arc A centered at $(u^0, -\epsilon)$ that goes from $(u^0 + r, v^0 - r)$ to $(u^0 - r, v^0 - r)$ and its range $f(A)$ where f is as in (2.15). Figure 2.1b shows that the bounding convex polygon of $f(A)$ contains the origin, and therefore $\bar{B}(x^0, r)$ fails the convex bounding polygon test.

2.6 Time complexity analysis

In this section, we prove a number of theorems relating to the behavior of the KTS-LS algorithm. We analyze the efficiency of KTS-LS by showing that a patch either is a subset of a safe region, passes the Kantorovich test, or passes the exclusion test when it is smaller than a certain size that depends on the condition number of the function. Hence, we have the upper bound of the total number of patches examined by KTS-LS in order to solve the intersection problem.

Recall that the Lipschitz constant $\hat{\omega}$ given by (2.12) is not the smallest Lipschitz constant of $f'(x^0)^{-1}f$ over D', where D' is given by (2.9). However, we can show that $\hat{\omega} \leq 4\theta\omega$, where ω denotes the smallest Lipschitz constant of $f'(x^0)^{-1}f$ over D'. Since $\hat{\omega}$ is computed from the endpoints of the bounding intervals of $\max_{x\in[l,h]^2} |\hat{g}'_{ijk}(x)|$, by (2.5),

$$\begin{aligned} \hat{\omega} &\leq 4\theta \max_{i,j,k} \max_{x\in[l,h]^2} \left|\hat{g}'_{ijk}(x)\right| \\ &= 4\theta \max_{i,j,k} \max_{x\in X} \left|g'_{ijk}(x)\right| \\ &\leq 4\theta \max_{x\in X} \|g'(x)\| = 4\theta\omega. \end{aligned} \tag{2.16}$$

With this bound on $\hat{\omega}$, we can now analyze the behavior of the Kantorovich test.

Theorem 2.6.1. *Let x^0 be a point in $[0,1]^2$ such that $f'(x^0)$ is invertible. Let x^* be a zero of f that is contained in $\bar{B}(x^0, r)$, where r is the radius of the patch under consideration. The patch $X = \bar{B}(x^0, r) \subseteq [0,1]^2$ passes the Kantorovich test if*

$$r \leq \frac{\gamma(\theta) - 1}{\gamma(\theta)\omega_{D'}}. \tag{2.17}$$

Proof. The first step is to show that $\eta\hat{\omega} \leq 1/4$, where $\hat{\omega}$ is as in (2.12). Since

$r \leq 1/2$, $\bar{B}(x^0, 2\gamma(\theta)r) \subseteq D'$. Observe that for any $x, y \in D'$,

$$\begin{aligned}
\left\|f'(x^0)^{-1}(f'(x) - f'(y))\right\| &= \| \left(f'(x^*)^{-1} + (f'(x^0)^{-1} - f'(x^*)^{-1})\right) \\
&\quad (f'(x) - f'(y)) \| \\
&\leq \left\|f'(x^*)^{-1}\left(f'(x) - f'(y)\right)\right\| + \|f'(x^*)^{-1} \\
&\quad (f'(x^*) - f'(x^0))f'(x^0)^{-1}(f'(x) - f'(y))\| \\
&\leq \omega_{D'} \|x - y\| + \left\|f'(x^*)^{-1}(f'(x^*) - f'(x^0))\right\| \cdot \\
&\quad \left\|f'(x^0)^{-1}(f'(x) - f'(y))\right\| \\
&\leq \omega_{D'} \|x - y\| + \\
&\quad \omega_{D'} \left\|x^* - x^0\right\| \cdot \left\|f'(x^0)^{-1}(f'(x) - f'(y))\right\| \\
&\leq \omega_{D'} \|x - y\| + \\
&\quad \omega_{D'} r \cdot \left\|f'(x^0)^{-1}(f'(x) - f'(y))\right\|. \qquad (2.18)
\end{aligned}$$

Since (2.17) implies

$$1 - \omega_{D'} r \geq 1/\gamma(\theta) > 0, \qquad (2.19)$$

the inequality (2.18) becomes

$$\left\|f'(x^0)^{-1}(f'(x) - f'(y))\right\| \leq \left(\frac{\omega_{D'}}{1 - \omega_{D'} r}\right) \|x - y\|.$$

Hence

$$\omega \leq \frac{\omega_{D'}}{1 - \omega_{D'} r}, \qquad (2.20)$$

where ω is the smallest Lipschitz constant of $f'(x^0)^{-1} f'$ over D'.

Recall that $f(x^*) = 0$ and $X \subseteq D'$. Observe that

$$\begin{aligned}
\eta &\equiv \left\| f'(x^0)^{-1} f(x^0) \right\| \\
&= \left\| f'(x^0)^{-1} (f(x^0) - f(x^*)) \right\| \\
&\leq \left(\max_{x \in X} \left\| f'(x^0)^{-1} f'(x) \right\| \right) \cdot \left\| x^0 - x^* \right\| \\
&\leq \left(\max_{x \in X} \left\| f'(x^0)^{-1} (f'(x) - f'(x^0)) + f'(x^0)^{-1} f'(x^0) \right\| \right) \cdot r \\
&\leq \left(\max_{x \in X} \left\| f'(x^0)^{-1} (f'(x) - f'(x^0)) \right\| + 1 \right) \cdot r \\
&\leq (\omega r + 1) r. \qquad (2.21)
\end{aligned}$$

Using (2.16), (2.17), (2.19), (2.20), and (2.21) together give

$$\eta \hat{\omega} \leq \frac{1}{4}.$$

The last step is to to verify the other condition that $\bar{B}(x^0, \rho_-) \subseteq \bar{B}(x^0, 2\gamma(\theta) r)$. Noting that $\sqrt{1-2h} \geq 1 - 2h$ for $0 \leq h \leq 1/2$, it is seen that

$$\begin{aligned}
\rho_-(\eta, \hat{\omega}) &= \frac{1 - \sqrt{1 - 2\eta\hat{\omega}}}{\hat{\omega}} \\
&\leq 2\eta \\
&\leq 2(\omega r + 1) r \\
&\leq 2\gamma(\theta) r.
\end{aligned}$$

□

Next results are concerned with the size of the patch satisfying the exclusion test.

Lemma 2.6.2. *Let $f : \mathbb{C}^n \to \mathbb{C}^n$ be a polynomial function with generic coefficients. Assume that the Jacobians at all zeros of f are invertible. Let x^0 be a point in $\mathbb{R}^n$.*

If

$$\left\|f'(x^*)^{-1}f(x^0)\right\| \leq \frac{1}{2\omega_f} \tag{2.22}$$

for all complex zeros x^* of f, then there exists $\hat{x}^*$, a zero of f, such that

$$\begin{aligned} \left\|x^0 - \hat{x}^*\right\| &\leq \frac{1-\sqrt{1-2\omega_f \left\|f'(\hat{x}^*)^{-1}f(x^0)\right\|}}{\omega_f} \\ &\equiv \sigma(\hat{x}^*, x^0). \end{aligned} \tag{2.23}$$

Proof. By the assumption that f has generic coefficients, the polynomial f has a finite number of zeros. Let x_1^*, x_2^*, ..., x_d^* be all the complex zeros of f. Recall that a multiple zero has singular Jacobian. Hence, f has no multiple zeros by assumption.

Define the polynomial $\bar{f}(x) = f(x) - f(x^0)$. Note that x^0 is a zero of $\bar{f}$. We apply Kantorovich's theorem for complex functions (see [9]) to each x_i^* with respect to $\bar{f}$. For each x_i^*, we use $D = \bar{B}(x_i^*, \rho_*(x_i^*))$ and $\omega = \omega_f$. Since $\eta \equiv \left\|\bar{f}'(x_i^*)^{-1}\bar{f}(x_i^*)\right\| = \|f'(x_i^*)^{-1}f(x^0)\|$, the assumption (2.22) guarantees that the condition $\eta\omega \leq 1/2$ is satisfied. The condition $\bar{B}(x_i^*, \rho_-) \subseteq D$ is also satisfied by the definition of D. Therefore, Kantorovich's theorem states that there is a zero of $\bar{f}$, call it $\bar{x}_i^*$, such that

$$\|\bar{x}_i^* - x_i^*\| \leq \sigma(x_i^*, x^0). \tag{2.24}$$

Recall that, for any j, x_j^* is the unique zero of f in $\bar{B}(x_j^*, \rho_*(x_j^*))$. Therefore,

$$\left\|x_i^* - x_j^*\right\| > \max\{\rho_*(x_i^*), \rho_*(x_j^*)\}, i \neq j. \tag{2.25}$$

But (2.24) and (2.25) together imply that

$$\bar{x}_i^* \neq \bar{x}_j^*, i \neq j. \tag{2.26}$$

Hence the mapping $x_i^* \to \bar{x}_i^*$ is injective. But since f has generic coefficients and f and $\bar{f}$ are of the same degrees, f has at least as many zeros as $\bar{f}$ [7]. This implies that $x^0 = \bar{x}_i^*$, for some i. The lemma follows. □

Theorem 2.6.3. *Let $f(x) = f(u, v)$ be a polynomial system in basis $\phi_i(u)\phi_j(v)$ in two dimensions with generic coefficients. Let $x^0 = (u^0, v^0)$ be a point in $[0, 1]^2$ such that $f'(x^0)$ is invertible and $f(x^0) \neq 0$, x^* be the closest zero in $\mathbb{R}^2$ of f to x^0, and δ denote $\|x^0 - x^*\|$. Let $r > 0$ be such that $\bar{B}(x^0, r) \subseteq [0, 1]^2$. Assume $\delta > \frac{1}{\omega_f}$. Define $\hat{f}(\hat{u}, \hat{v})$ such that*

$$\hat{f}(\hat{u}, \hat{v}) = f(\frac{2r}{h-l}\hat{u} - \frac{2hr}{h-l} + u^0 + r, \frac{2r}{h-l}\hat{v} - \frac{2hr}{h-l} + v^0 + r). \tag{2.27}$$

In other word, $\hat{f}$ is a polynomial in basis $\phi_i(u)\phi_j(v)$ that reparametrizes with $[l, h]^2$ the surface defined by f over the patch $\bar{B}(x^0, r)$. The bounding polygon of $\{\hat{f}(u, v) : l \leq u, v \leq h\}$ satisfying item 2 of the basis properties listed in Section 2.2 does not contain the origin if

$$r \leq \frac{1}{2\theta \operatorname{cond}(f)^2}. \tag{2.28}$$

Proof. Let X denote the patch $\bar{B}(x^0, r)$ and x denote an arbitrary point in X. Since $\delta > \frac{1}{\omega_f}$, the contrapositive of Lemma 2.6.2 implies there exists a zero $\bar{x}^*$ of f satisfying $\|f'(\bar{x}^*)^{-1} f(x^0)\| > \frac{1}{2\omega_f}$. Therefore, the condition (2.28) implies

$$\begin{aligned} r &\leq \frac{1}{2\theta \operatorname{cond}(f)^2} \\ &\leq \frac{1}{2\theta\omega_f \|f'(\bar{x}^*)^{-1} f'(x)\|} \\ &< \frac{\|f'(\bar{x}^*)^{-1} f(x^0)\|}{\theta \|f'(\bar{x}^*)^{-1} f'(x)\|}. \end{aligned}$$

More specifically, we have

$$r < \frac{\|f'(\bar{x}^*)^{-1} f(x^0)\|}{\theta \max_{y \in X} \|f'(\bar{x}^*)^{-1} f'(y)\|},$$

which is equivalent to

$$\theta \cdot \max_{y \in X} \left\| f'(\bar{x}^*)^{-1} f'(y) \right\| \cdot r < \left\| f'(\bar{x}^*)^{-1} f(x^0) \right\|. \tag{2.29}$$

Recall that $\max_{y\in X}\|f'(\bar{x}^*)^{-1}f'(y)\|$ is the Lipschitz constant for $f'(\bar{x}^*)^{-1}f$ on X. Hence, for any $x \in X$,

$$\begin{aligned}\left\|f'(\bar{x}^*)^{-1}f(x) - f'(\bar{x}^*)^{-1}f(x^0)\right\| &\leq \max_{y\in X}\left\|f'(\bar{x}^*)^{-1}f'(y)\right\| \cdot \left\|x - x^0\right\| \\ &\leq \max_{y\in X}\left\|f'(\bar{x}^*)^{-1}f'(y)\right\| \cdot r. \end{aligned} \tag{2.30}$$

Combining (2.29) and (2.30) gives

$$\theta \cdot \left\|f'(\bar{x}^*)^{-1}f(x) - f'(\bar{x}^*)^{-1}f(x^0)\right\| < \left\|f'(\bar{x}^*)^{-1}f(x^0)\right\|,$$

which is equivalent to

$$\theta \cdot \left\|f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}) - f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}^0)\right\| < \left\|f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}^0)\right\|$$

for some $\hat{x} \in [l,h]^2$, where $\hat{x}$ is the rescaled x and $\hat{x}^0$ is the rescaled x^0 according to (2.27). In particular,

$$\theta \cdot \max_{\hat{x}\in[l,h]^2}\left\|f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}) - f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}^0)\right\| < \left\|f'(\bar{x}^*)^{-1}\hat{f}(\hat{x}^0)\right\|. \tag{2.31}$$

Let $h(\hat{x}) \equiv f'(\bar{x}^*)^{-1}\hat{f}(\hat{x})$ and $g(\hat{x}) \equiv h(\hat{x}) - h(\hat{x}^0)$. By (2.5),

$$\|z\| \leq \theta \cdot \max_{\hat{x}\in[l,h]^2}\|g(\hat{x})\|, \tag{2.32}$$

for any z in the bounding polygon P_g of $\{g(\hat{x}) : \hat{x} \in [l,h]^2\}$. Since the bounding polygon is required to be translationally invariant (item 2 of the basis properties listed in Section 2.2), (2.32) is equivalent to

$$\left\|y - h(\hat{x}^0)\right\| \leq \theta \cdot \max_{\hat{x}\in[l,h]^2}\left\|h(\hat{x}) - h(\hat{x}^0))\right\|, \tag{2.33}$$

for any y in the bounding polygon P_h of $\{h(\hat{x}) : \hat{x} \in [l,h]^2\}$. Substituting (2.33) into the left hand side of (2.31) yields

$$\left\|y - h(\hat{x}^0)\right\| < \left\|h(\hat{x}^0)\right\|,$$

which implies that P does not contain the origin. Since $f'(\bar{x}^*)^{-1}$ is invertible and the bounding polygon is affinely invariant, the bounding polygon of $\{\hat{f}(\hat{x}) : \hat{x} \in [l,h]^2\}$ does not contain the origin, either. □

Theorem 2.6.4. *Let $f(x) = f(u, v)$ be a polynomial system in basis $\phi_i(u)\phi_j(v)$ in two dimensions with generic coefficients whose zeros are sought. Let $X = \bar{B}(x^0, r)$ be a patch under consideration during the course of the KTS-LS algorithm. The algorithm does not need to subdivide X if*

$$r \leq \frac{1}{2} \cdot \min \left\{ \frac{1 - 1/\gamma(\theta)}{\omega_f}, \frac{1}{2\theta \operatorname{cond}(f)^2} \right\}. \tag{2.34}$$

Proof. If $\delta > 1/\omega_f$, where δ is the distance between x^0 and the closest zero x^*, $r \leq (1 - 1/\gamma(\theta))/(2\omega_f)$ implies that X does not contain a zero. Therefore, $r \leq 1/(4\theta \operatorname{cond}(f)^2)$ implies that X is excluded by the exclusion test according to Theorem 2.6.3.

Observe that $\omega_* \leq \omega_f$. If $\delta \leq 1/\omega_f$, for any $x \in X$,

$$\begin{aligned} \|x - x^*\| &\leq \left\|x - x^0\right\| + \left\|x^0 - x^*\right\| \\ &\leq r + \delta \\ &\leq \frac{1 - 1/\gamma(\theta)}{\omega_f} + \frac{1}{\omega_f} \\ &< \frac{2}{\omega_f} \leq \frac{2}{\omega_*} = \rho_*. \end{aligned}$$

In other word, X is contained within $B(x^*, \rho_*)$, a safe region and therefore is excluded, provided that x^* is found before X is checked against all safe regions. By Theorem 2.6.1, x^* is found by a region of size $2r \leq (1-1/\gamma(\theta))/\omega_{D'}$. Since KTS-LS examines larger regions before smaller ones, x^* is found before X is checked against safe regions. □

It should be noted that

$$1 - 1/\gamma(\theta) \geq 1/(18\theta) \tag{2.35}$$

hence both terms of the right hand side of (2.34) are asymptotically linear in $1/\theta$. The inequality (2.35) follows from the fact that

$$\sqrt{1+a} \leq 1 + a/2 - a^2/9 \tag{2.36}$$

for any $a \in [0, 1/4]$. To prove (2.36), simplify (2.36) to $a^2 - 9a + 9/4 \geq 0$, whose left hand side is a convex quadratic polynomial that crosses the x-axis at $(9 - 6\sqrt{2})/2 \approx .2574$ and at $(9 + 6\sqrt{2})/2$.

2.7 Computational results

The KTS-LS algorithm is implemented in Matlab and is tested against a number of problem instances with varying condition numbers. As Bézier surfaces are widely used in geometric modeling, we choose to implement KTS-LS for the Bernstein basis case. Most of the test problems are created by using normally distributed random numbers as the coefficients c_{ij}'s of f. For some of the test problems especially those with high condition number, some coefficients are manually entered. The degrees of the test polynomials are between biquadratic and biquartic. As an example, the test case with $\text{cond}(f) = 3.5 \times 10^3$ is $c_{00} = (1.2, .5)^T$, $c_{01} = (-.6, -.6)^T$, $c_{02} = (.1, 1.1)^T$, $c_{10} = (-1.1, -.3)^T$, $c_{11} = (.6, -2.3)^T$, $c_{12} = (-2, -.1)^T$, $c_{20} = (.6, 1.2)^T$, $c_{21} = (-1.1, -1.2)^T$, and $c_{22} = (-.5, .4)^T$. This is the test problem for the result in the second row of Table 2.2.

For the experiment, we use the algorithm by Jónsson and Vavasis [15] to compute the complex zeros required to estimate the condition number. Table 2.2 compares the efficiency of KTS-LS with its condition number. The total number of subpatches examined by KTS-LS during the entire computation, the width of the smallest patch among those examined, and the maximum number of Newton

Table 2.2: Efficiency of KTS-LS algorithm on problems of different condition numbers.

cond(f)	Num. of zeros	Distance between two closest zeros	Num. of patches examined	Smallest width	Max. num. of Newton iterations
6.0×10^2	1	-	21	.0625	3
3.5×10^3	2	.4196	29	.0625	3
8.3×10^4	2	.6638	33	.0625	3
1.6×10^5	1	-	41	.03125	4
2.2×10^7	3	.3624	57	.03125	4
1.3×10^8	4	.2806	81	.015625	6
1.9×10^9	4	.3069	69	.03125	6
2.0×10^{10}	2	.7810	105	.015625	6
2.9×10^{11}	1	-	257	.0039	9

iterations (in the cases with more than one zero) to converge to a zero are reported. The result shows that KTS-LS needs to examine more number of patches and needs to subdivide to smaller patches as the condition number becomes larger. Note that the high number of Newton iterations of some test cases is due to roundoff error.

2.8 Summary

We present the KTS-LS algorithm for finding the intersections between a parametric surface and a line. By using a combination of subdivision and Kantorovich's theorem, our algorithm can take advantage of the quadratic convergence of Newton's method without the problems of divergence and missing some intersections that commonly occur with Newton's method. KTS-LS can operate on polynomials in any bases satisfying the properties listed in Section 2.2. The power, Bernstein, and first-kind Chebyshev bases are examples of such bases. We also show that the efficiency of KTS-LS has an upper bound that depends solely on the condi-

tioning of the problem and the constant depending on the basis representing the polynomials.

Chapter 3

Properties of Polynomial Bases used in a Line/Surface Intersection Algorithm

In Chapter 2, it is shown that the proposed Algorithm KTS-LS for solving line/surface intersections can operate on polynomials represented in any basis satisfying the properties listed in Section 2.2. In addition, the running time of KTS-LS is shown to depend on the type of bounding polygons that can be computed for the chosen basis. This chapter investigates the power, Bernstein, and first-kind Chebyshev bases, proves the claim from Chapter 2 that these three bases satisfy the required basis properties for them to be compatible with KTS-LS, and derives the constants θ's for each of the bases and their corresponding bounding polygons. By comparing these constants, we can determine which basis results in the smallest upper bound on the running time of KTS-LS.

3.1 Properties of the power, Bernstein, and Chebyshev bases

Recall from Chapter 2 that the line/surface intersection problem can be reduced to the problem of finding all zeros of

$$f(u,v) \equiv \sum_{i=0}^{m}\sum_{j=0}^{n} c_{ij}\phi_i(u)\phi_j(v), \quad 0 \le u,v \le 1, \tag{3.1}$$

where $c_{ij} \in \mathbb{R}^2$ $(i = 0,1,\ldots,m; j = 0,1,\ldots,n)$ denote the coefficients. As mentioned above, the basis used to represent the polynomial system (3.1) must satisfy the properties listed in Section 2.2 for KTS-LS to work efficiently. Three bases,

the power, Bernstein, and Chebyshev bases are examined in detail. Recall that the power basis for polynomials of degree n is $\phi_k(t) = t^k$ $(0 \leq k \leq n)$. The Bernstein basis is $\phi_k(t) = Z_{k,n}(t) = \begin{pmatrix} n \\ k \end{pmatrix} (1-t)^{n-k} t^k$ $(0 \leq k \leq n)$. The Chebyshev basis is $\phi_k(t) = T_k(t)$ $(0 \leq k \leq n)$, where $T_k(t)$ is the Chebyshev polynomial of the first kind generated by the recurrence relation

$$\begin{aligned} T_0(t) &= 1, \\ T_1(t) &= t, \\ T_{k+1}(t) &= 2tT_k(t) - T_{k-1}(t) \text{ for } k \geq 1. \end{aligned} \tag{3.2}$$

Another way to define the Chebyshev polynomials of the first kind is through the identity

$$T_k(\cos\alpha) = \cos k\alpha. \tag{3.3}$$

This second definition shows, in particular, that all zeros of $T_k(t)$ lie in $[-1, 1]$. It also shows that $-1 \leq T_k(t) \leq 1$ for any $-1 \leq t \leq 1$.

3.1.1 Bounding polygons

The choices of l and h and the definitions of bounding polygons of the surface $S = \{f(u,v) : l \leq u, v \leq h\}$, where $f(u,v)$ is represented by one of the three bases, that satisfy the required properties are as follows: For Bernstein basis, the convex hull of the coefficients (control points), call it P_1, satisfies the requirements for $l = 0$ and $h = 1$. The convex hull P_1 can be described as

$$P_1 = \left\{ \sum_{i,j} c_{ij} s_{ij} : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\}.$$

For power and Chebyshev bases, the bounding polygon

$$P_2 = \left\{ c_{00} + \sum_{i+j>0} c_{ij} s_{ij} : -1 \leq s_{ij} \leq 1 \right\}$$

satisfies the requirements for $l = -1$ and $h = 1$. Note that P_2 is a bounding polygon of S in the Chebyshev case since $|T_k(t)| \leq 1$ for any $k \geq 0$ and any $t \in [-1, 1]$. Determining whether $0 \in P_2$ is done by solving a small linear programming problem. To determine if $0 \in P_1$, the convex hull is constructed by conventional method and is tested to see if it contains the origin.

The affine and translational invariance of P_1 and P_2 for their respective bases can be verified as follows: Let

$$g(u, v) = Af(u, v) + b = \sum_{i=0}^{m} \sum_{j=0}^{n} c'_{ij} \phi_i(u) \phi_j(v).$$

For the Bernstein basis, by using the property that $\sum_{k=0}^{n} Z_{k,n}(t) = 1$, it is seen that $c'_{ij} = Ac_{ij} + b$ for all c_{ij}'s. Therefore, the bounding polygon of $\{g(u, v) : 0 \leq u, v \leq 1\}$ is

$$\begin{aligned}
P'_1 &= \left\{ \sum_{i,j} c'_{ij} s_{ij} : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\} \\
&= \left\{ \sum_{i,j} (Ac_{ij} + b) s_{ij} : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\} \\
&= \left\{ A \sum_{i,j} c_{ij} s_{ij} + b \sum_{i,j} s_{ij} : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\} \\
&= \left\{ A \sum_{i,j} c_{ij} s_{ij} + b : \sum_{i,j} s_{ij} = 1, 0 \leq s_{ij} \leq 1 \right\} \\
&= \{ Ax + b : x \in P_1 \} .
\end{aligned}$$

For the power and the Chebyshev bases, note that $\phi_0(u)\phi_0(v) = 1$ for both bases. Hence, $c'_{00} = Ac_{00} + b$ and $c'_{ij} = Ac_{ij}$ for $i + j > 0$. The bounding polygon

of $\{g(u,v) : 0 \leq u, v \leq 1\}$ for this case is

$$\begin{aligned} P_2' &= \left\{ c_{00}' + \sum_{i+j>0} c_{ij}' s_{ij} : -1 \leq s_{ij} \leq 1 \right\} \\ &= \left\{ Ac_{00} + b + \sum_{i+j>0} Ac_{ij} s_{ij} : -1 \leq s_{ij} \leq 1 \right\} \\ &= \left\{ A\left(c_{00} + \sum_{i+j>0} c_{ij} s_{ij} \right) + b : -1 \leq s_{ij} \leq 1 \right\} \\ &= \{ Ax + b : x \in P_2 \} . \end{aligned}$$

3.1.2 The size of the bounding polygons compared to the size of the bounded surface

Item 2c of the basis properties in effect ensures that the bounding polygons are not unboundedly larger than the actual surface itself lest the bounding polygons lose their usefulness. The value θ also can be used as a measure of the tightness of the bounding polygon. Recall from Theorem 2.6.4 that the efficiency of KTS-LS depends on θ.

Since the bounding polygons P_1 and P_2 are defined by the coefficients of f, our approach to derive θ is to first derive ξ, a function of m and n, satisfying

$$\|c_{ij}\| \leq \xi \max_{l \leq u,v \leq h} \|f(u,v)\| ,$$

for any coefficient c_{ij} of f. But the following lemma shows that one needs only derive the equivalent of ξ for univariate polynomial to derive ξ itself.

Lemma 3.1.1. *Assume there exists a function $h(n)$ such that*

$$\|b_i\| \leq h(n) \max_{l \leq t \leq h} \|g(t)\| \tag{3.4}$$

for any b_i $(i = 0, 1, \ldots, n)$*, and any univariate polynomial* $g(t) = \sum_{i=0}^{n} b_i \phi_i(t)$*. Then*

$$\|c_{ij}\| \leq h(m)h(n) \max_{l \leq u,v \leq h} \|f(u,v)\|, \tag{3.5}$$

for any c_{ij} $(i = 0, 1, \ldots, m;\ j = 0, 1, \ldots, n)$*, and any bivariate polynomial* $f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij} \phi_i(u) \phi_j(v)$*.*

Proof. For any $j = 0, 1, \ldots, n$, define a univariate polynomial $g_j(u) = \sum_{i=0}^{m} c_{ij} \phi_i(u)$. Let $u^0 = \arg\max_{l \leq u \leq h} \|\sum_{i=0}^{m} c_{ij} \phi_i(u)\|$. By applying (3.4) to $g_j(u)$,

$$\begin{aligned} \|c_{ij}\| &\leq h(m) \max_{l \leq u \leq h} \left\| \sum_{i=0}^{m} c_{ij} \phi_i(u) \right\| \\ &= h(m) \left\| \sum_{i=0}^{m} c_{ij} \phi_i(u^0) \right\|. \end{aligned} \tag{3.6}$$

Define another univariate polynomial $\hat{g}(v) = \sum_{j=0}^{n} \left(\sum_{i=0}^{m} c_{ij} \phi_i(u^0) \right) \phi_j(v)$. By applying (3.4) to $\hat{g}(v)$, (3.6) becomes

$$\begin{aligned} \|c_{ij}\| &\leq h(m)h(n) \max_{l \leq v \leq h} \left\| \sum_{j=0}^{n} \left(\sum_{i=0}^{m} c_{ij} \phi_i(u^0) \right) \phi_j(v) \right\| \\ &\leq h(m)h(n) \max_{l \leq u,v \leq h} \left\| \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij} \phi_i(u) \phi_j(v) \right\| \\ &= h(m)h(n) \max_{l \leq u,v \leq h} \|f(u,v)\|. \end{aligned}$$

□

We proceed to derive ξ of the three bases, starting with the Bernstein basis. The following lemma regarding the product of two polynomials in Bernstein basis is needed to find ξ for Bernstein case.

Lemma 3.1.2. *Let*

$$f(t) = \sum_{i=0}^{n} c_i Z_{i,n}(t), \quad 0 \leq t \leq 1$$

and

$$g(t) = \sum_{i=0}^{n'} c'_i Z_{i,n'}(t), \quad 0 \leq t \leq 1.$$

Then

$$f(t)g(t) = \sum_{i=0}^{n+n'} b_i Z_{i,n+n'}(t),$$

where

$$|b_i| \leq \max_i |c_i| \cdot \max_i |c'_i|.$$

Proof. Straightforward arithmetic shows that

$$b_i = \sum_{k=\max(0,i-n')}^{\min(n,i)} \frac{\begin{pmatrix} n \\ k \end{pmatrix} \begin{pmatrix} n' \\ i-k \end{pmatrix}}{\begin{pmatrix} n+n' \\ i \end{pmatrix}} c_k c'_{i-k}.$$

Taking absolute value on both sides and bounding $|c_k|$ (resp. $|c'_{i-k}|$) with $\max_i |c_i|$ (resp. $\max_i |c'_i|$) gives

$$|b_i| \leq \max |c_i| \cdot \max |c'_i| \sum_{k=\max(0,i-n')}^{\min(n,i)} \frac{\begin{pmatrix} n \\ k \end{pmatrix} \begin{pmatrix} n' \\ i-k \end{pmatrix}}{\begin{pmatrix} n+n' \\ i \end{pmatrix}}.$$

Recall the combinatorial identity

$$\begin{pmatrix} n+n' \\ i \end{pmatrix} = \sum_{k=\max(0,i-n')}^{\min(n,i)} \begin{pmatrix} n \\ k \end{pmatrix} \begin{pmatrix} n' \\ i-k \end{pmatrix}.$$

Hence, the lemma follows. □

With the above lemma, we are ready to derive ξ of the Bernstein basis.

Theorem 3.1.3. *Let $f(t)$ be a polynomial system*

$$f(t) \;\; = \;\; \textstyle\sum_{i=0}^{n} c_i Z_{i,n}(t), \quad 0 \leq t \leq 1,$$

where $c_i \in \mathbb{R}^d$. The norm of the coefficients can be bounded by

$$\|c_i\| \leq \xi_B(n) \max_{t:0\leq t\leq 1} \|f(t)\|, \tag{3.7}$$

where

$$\xi_B(n) = \sum_{i=0}^{n} \prod_{j=0,1,\ldots,i-1,i+1,\ldots,n} \frac{\max\{|n-j|,|j|\}}{|i-j|} = O(n^{n+1}).$$

Remark. An inequality in the other direction, namely, that

$$\max_{t:0\leq t\leq 1} \|f(t)\| \leq \max \|c_i\|,$$

is a well-known consequence of the convex hull property of Bernstein polynomials [8].

Proof. By definition of infinity norm, it suffices to prove the lemma for the case $c_i \in \mathbb{R}$. Therefore, it is assumed that $d = 1$ for the rest of this proof.

Let $t_j = j/n$ $(j = 0, 1, \ldots, n)$. Define a matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$ having element

$$A_{j+1,i+1} = Z_{i,n}(t_j).$$

Define the vectors $c = (c_0, c_1, \ldots, c_n)^T$ and $f = (f(t_0), f(t_1), \ldots, f(t_n))^T$. Observe that

$$Ac = f. \tag{3.8}$$

We claim that A is invertible. In particular, we show that the linear system $Ax = b$ has solution for any arbitrary $b \in \mathbb{R}^{n+1}$. Due to the definition of A, solving the system $Ax = b$ is equivalent to finding the coefficients of the polynomial

$$g(t) = \sum_{i=0}^{n} x_{i+1} Z_{i,n}(t) \tag{3.9}$$

with the property that $g(t_0) = b_1$, $g(t_1) = b_2$, $\ldots$, $g(t_n) = b_{n+1}$. The polynomial g satisfying such property is the Lagrange interpolant

$$g(t) = \sum_{j=0}^{n} \left(b_{j+1} \prod_{j'=0,\ldots,j-1,j+1,\ldots,n} \frac{t - t_{j'}}{t_j - t_{j'}} \right). \tag{3.10}$$

Transforming (3.10) to the Bernstein basis yields the solution x.

Knowing that A is invertible, we multiply both sides of (3.8) by A^{-1},

$$c = A^{-1} f, \tag{3.11}$$

and hence, for any $i = 0, 1, \ldots, n$,

$$\begin{aligned} |c_i| &\leq \|c\| \\ &\leq \left\|A^{-1}\right\| \cdot \|f\| \\ &\leq \left\|A^{-1}\right\| \cdot \max_{t:0\leq t\leq 1} |f(t)| . \end{aligned} \tag{3.12}$$

Comparing (3.7) to (3.12), it is seen that the final step is to show that $\|A^{-1}\| \leq \xi_B(n)$.

Observe that the ith column of A^{-1} is $A^{-1}e_i$, where e_i denotes the ith column of the identity matrix. Let $g_i(t)$ be a polynomial in the Bernstein basis and let $\{c'_{i'}\}$ be its coefficients. With similar reasoning as the above,

$$\begin{pmatrix} c'_0 \\ \vdots \\ c'_n \end{pmatrix} = A^{-1} \begin{pmatrix} g_i(t_0) \\ \vdots \\ g_i(t_n) \end{pmatrix}. \tag{3.13}$$

But (3.13) implies that the ith column of A^{-1}, $A^{-1}e_i$, are the coefficients of g_i such that, for $j = 0, 1, \ldots, n$,

$$g_i(t_j) = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases} \tag{3.14}$$

The following Lagrange interpolant g_i satisfies (3.14):

$$\begin{aligned} g_i(t) &= \prod_{j=0,\ldots,i-1,i+1,\ldots,n} \frac{t-t_j}{t_i-t_j} \\ &= \prod_{j=0,\ldots,i-1,i+1,\ldots,n} \left(\frac{n-j}{i-j} t - \frac{j}{i-j}(1-t) \right). \end{aligned} \tag{3.15}$$

Note that each term of the product in (3.15) is a polynomial in Bernstein basis with coefficients $(n-j)/(i-j)$ and $j/(i-j)$. Applying Lemma 3.1.2 to (3.15) shows that

$$\left\| A^{-1} e_i \right\| \leq \prod_{j=0,1,\ldots,i-1,i+1,\ldots,n} \frac{\max\{|n-j|, |j|\}}{|i-j|}. \tag{3.16}$$

Since (3.16) holds for any column i of A^{-1}, the lemma follows. □

Next is the derivation of ξ of the Chebyshev basis. The following identity is useful for this derivation:

$$\sum_{k=1}^{n} T_i(t_k) T_j(t_k) = \begin{cases} 0 & i \neq j \\ n & i = j = 0 \\ n/2 & i = j \neq 0, \end{cases} \tag{3.17}$$

for $i, j = 0, \ldots, n-1$, where t_k $(k = 1, 2, \ldots, n)$ are the n zeros of $T_n(t)$.

Theorem 3.1.4. *Let $f(t)$ be a polynomial system*

$$f(t) = \sum_{i=0}^{n} c_i T_i(t),$$

where $c_i \in \mathbb{R}^d$. The norm of the coefficients can be bounded by

$$\|c_i\| \leq \sqrt{2} \max_{t:-1\leq t\leq 1} \|f(t)\|. \tag{3.18}$$

Proof. By definition of infinity norm, it suffices to prove the lemma for the case $c_i \in \mathbb{R}$. Therefore, it is assumed that $d = 1$ for the rest of this proof.

Let t_j $(j = 1, 2, \ldots, n+1)$ be the $n+1$ zeros of $T_{n+1}(t)$, which lie in $[-1, 1]$. Define a matrix $A \in \mathbb{R}^{(n+1)\times(n+1)}$ having element

$$A_{j+1,i+1} = T_i(t_j).$$

Define the vectors $c = (c_0, c_1, \ldots, c_n)^T$ and $f = (f(t_0), f(t_1), \ldots, f(t_n))^T$. Observe that

$$Ac = f. \tag{3.19}$$

By (3.17),

$$A^T A = \operatorname{diag}\left(n+1, (n+1)/2, (n+1)/2, \ldots, (n+1)/2\right),$$

which implies that A is invertible and

$$A^{-1}A^{-T} = \operatorname{diag}\left(1/(n+1), 2/(n+1), 2/(n+1), \ldots, 2/(n+1)\right). \tag{3.20}$$

The equation (3.20) implies

$$\left\|A^{-1}\right\|_2 = \sqrt{2/(n+1)}. \tag{3.21}$$

Finally, from (3.19) and (3.21),

$$\begin{aligned}
|c_i| &\leq \|c\|_2 \\
&\leq \left\|A^{-1}\right\|_2 \|f\|_2 \\
&\leq \sqrt{n+1}\left\|A^{-1}\right\|_2 \|f\| \\
&\leq \sqrt{n+1}\left\|A^{-1}\right\|_2 \max_{t:-1\leq t\leq 1} |f(t)| \\
&= \sqrt{2} \max_{t:-1\leq t\leq 1} |f(t)|. \qquad \square
\end{aligned}$$

Last is the power basis. Our approach to derive ξ of power basis is to derive the relationship between the coefficients of a polynomial in power basis and the coefficients of the same polynomial but written in Chebyshev basis. By using this relationship and Theorem 3.1.4, ξ of the power basis can be computed.

Lemma 3.1.5. *Let f be a univariate polynomial such that*

$$f(t) = \sum_{i=0}^{n} a_i t^i = \sum_{i=0}^{n} c_i T_i(t).$$

In other words, $\{a_i\}$ are the coefficients of f when written in the power basis and $\{c_i\}$ are the coefficients of f when written in the Chebyshev basis. Then

$$|a_i| \leq \frac{3^{n+1}-1}{2} \max_{j=0,\ldots,n} |c_j|,$$

for any $i = 0, \ldots, n$.

Proof. Let $D = [d_{i,j}]$ be the $n+1$-by-$n+1$ matrix such that

$$a = Dc,$$

where $a = (a_0, a_1, \ldots, a_n)^T$ and $c = (c_0, c_1, \ldots, c_n)^T$. Note that

$$T_j(t) = \sum_{i=0}^{j} d_{i+1,j+1} t^i.$$

Recall the recurrence relation $T_j(t) = 2tT_{j-1}(t) - T_{j-2}(t)$. It follows from this recurrence that

$$|d_{i+1,j+1}| \leq 3^j. \tag{3.22}$$

That is, when $T_j(t)$ is written in the power basis, the resulting coefficients (of power basis) is less than or equal to 3^j. The inequality (3.22) can be verified by induction on the recurrence relation. Since the entries in the $(j+1)$th column of D is bounded by 3^j, we have, from geometric sum,

$$\|D\| \leq (3^{n+1}-1)/2.$$

The lemma follows from $\|a\| \leq \|D\| \, \|c\|$. □

Theorem 3.1.6. *Let $f(t)$ be a polynomial system*

$$f(t) = \sum_{i=0}^{n} c_i t^i,$$

where $c_i \in \mathbb{R}^d$. The norm of the coefficients can be bounded by

$$\|c_i\| \leq \frac{3^{n+1}-1}{\sqrt{2}} \max_{t:-1\leq t\leq 1} \|f(t)\|. \tag{3.23}$$

Proof. Follow directly from Theorem 3.1.4 and Lemma 3.1.5. □

Having ξ for each of the three bases, the values of θ for the three bases can now be derived.

Corollary 3.1.7. *Let*

$$f(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} c_{ij} Z_{i,m}(u) Z_{j,n}(v),$$

where $c_{ij} \in \mathbb{R}^2$ $(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$. Let P_1 be the convex hull of $\{c_{ij}\}$. Then, for any $y \in P_1$,

$$\|y\| \leq \xi_B(m)\xi_B(n) \max_{0\leq u,v\leq 1} \|f(u,v)\|.$$

Proof. By the convex hull property of Bernstein polynomials, $\|y\| \leq \max_{i,j} \|c_{ij}\|$ for any $y \in P_1$. The corollary then follows from Theorem 3.1.3 and Lemma 3.1.1. □

Corollary 3.1.8. *Let*

$$f(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} c_{ij} u^i v^j,$$

where $c_{ij} \in \mathbb{R}^2$ $(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$. Let

$$P_2 = \left\{ c_{00} + \sum_{i+j>0} c_{ij} s_{ij} : -1 \leq s_{ij} \leq 1 \right\}.$$

Then, for any $y \in P_2$,

$$\|y\| \leq \frac{(m+1)(n+1)(3^{m+1}-1)(3^{n+1}-1)}{2} \max_{-1\leq u,v\leq 1} \|f(u,v)\|.$$

Proof. For any $y \in P_2$,

$$\|y\| \leq \sum_{i=0}^{m} \sum_{j=0}^{n} \|c_{ij}\| .$$

The corollary then follows from Theorem 3.1.6 and Lemma 3.1.1. □

Corollary 3.1.9. *Let*

$$f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij} T_i(u) T_j(v),$$

where $c_{ij} \in \mathbb{R}^2$ $(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$. *Let*

$$P_2 = \left\{ c_{00} + \sum_{i+j>0} c_{ij} s_{ij} : -1 \leq s_{ij} \leq 1 \right\}.$$

Then, for any $y \in P_2$,

$$\|y\| \leq 2(m+1)(n+1) \max_{-1 \leq u,v \leq 1} \|f(u,v)\| .$$

Proof. For any $y \in P_2$,

$$\|y\| \leq \sum_{i=0}^{m} \sum_{j=0}^{n} \|c_{ij}\| .$$

The corollary then follows from Theorem 3.1.4 and Lemma 3.1.1. □

3.2 Relationship between the bounding polygon of the power basis and that of Chebyshev basis

Let P_2^p denote the bounding polygon P_2 computed from the power basis representation of a polynomial and P_2^c denote P_2 computed from the Chebyshev basis representation of it. The results from previous section show that the value θ of P_2^c is smaller than θ of P_2^p. This only implies that the worst case of P_2^c is better than the worst case of P_2^p. Comparing the values of θ's of the two does not indicate that

P_2^c is always a better choice than P_2^p for every polynomial. The following results show, however, that P_2^c is, in fact, always a better choice than P_2^p. Specifically, this section shows that for any given polynomial, its bounding polygon P_2^c is a subset of its bounding polygon P_2^p.

The following two lemmas show that when representing monomials t^k in Chebyshev basis, each coefficient is nonnegative, and the sum of all coefficients are exactly 1. These results are useful in relating P_2^p to P_2^c.

Lemma 3.2.1. *Let d_{ki}'s ($k = 0, 1, \ldots$; $i = 0, 1, \ldots, k$) be the numbers satisfying $t^k = \sum_{i=0}^{k} d_{ki} T_i(t)$. Then*

$$d_{ki} \geq 0,$$

for any $k = 0, 1, \ldots$ and any $i = 0, 1, \ldots, k$.

Proof. We prove the lemma by induction on k. The base cases $k = 0$ and $k = 1$ are trivial. For the inductive step, for any $k \geq 1$,

$$\begin{aligned} t^{k+1} &= t \cdot t^k \\ &= t \sum_{i=0}^{k} d_{ki} T_i(t) \\ &= \sum_{i=1}^{k} \frac{d_{ki}}{2} \left(2tT_i(t) - T_{i-1}(t)\right) + \sum_{i=1}^{k} \frac{d_{ki}}{2} T_{i-1}(t) + d_{k0} t T_0(t). \end{aligned}$$

By (3.2) and noting that $tT_0(t) = t = T_1(t)$,

$$\begin{aligned} t^{k+1} &= \sum_{i=1}^{k} \frac{d_{ki}}{2} T_{i+1}(t) + \sum_{i=1}^{k} \frac{d_{ki}}{2} T_{i-1}(t) + d_{k0} T_1(t) \\ &= \sum_{i=k}^{k+1} \frac{d_{k,i-1}}{2} T_i(t) + \sum_{i=2}^{k-1} \frac{d_{k,i-1} + d_{k,i+1}}{2} T_i(t) + \left(\frac{d_{k2}}{2} + d_{k0}\right) T_1(t) + \frac{d_{k1}}{2} T_0(t). \end{aligned}$$

Hence,

$$d_{k+1,i} = \begin{cases} d_{k,i-1}/2, & i = k, k+1, \\ \left(d_{k,i-1} + d_{k,i+1}\right)/2, & i = 2, \ldots, k-1, \\ d_{k2}/2 + d_{k0}, & i = 1, \\ d_{k1}/2, & i = 0. \end{cases} \tag{3.24}$$

But since $d_{ki} \geq 0$ for any $i = 0, \ldots, k$ by the induction hypothesis, (3.24) shows that $d_{k+1,i} \geq 0$ for any $i = 0, \ldots, k+1$. □

Lemma 3.2.2. *Let d_{ki}'s ($k = 0, 1, \ldots$; $i = 0, 1, \ldots, k$) be the numbers satisfying $t^k = \sum_{i=0}^{k} d_{ki} T_i(t)$. Then*

$$\sum_{i=0}^{k} d_{ki} = 1,$$

for any $k = 0, 1, \ldots$ and any $i = 0, 1, \ldots, k$.

Proof. We prove the lemma by induction on k. The base cases $k = 0$ and $k = 1$ are trivial. For the inductive step, the same reasoning as in the proof of Lemma 3.2.1 shows that $d_{k+1,i}$ is as in (3.24) for any $k \geq 1$. Therefore,

$$\begin{aligned} \sum_{i=0}^{k+1} d_{k+1,i} &= \sum_{i=k}^{k+1} \frac{d_{k,i-1}}{2} + \sum_{i=2}^{k-1} \frac{d_{k,i-1} + d_{k,i+1}}{2} + \left(\frac{d_{k2}}{2} + d_{k0}\right) + \frac{d_{k1}}{2} \\ &= \sum_{i=0}^{k} d_{ki} = 1, \end{aligned}$$

by the induction hypothesis. □

Theorem 3.2.3. *Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be a bivariate polynomial. Its bounding polygon P_2^c is a subset of its bounding polygon P_2^p.*

Proof. Let $f = f(u, v) = \sum_{k=0}^{m} \sum_{l=0}^{n} a_{kl} u^k v^l = \sum_{i=0}^{m} \sum_{j=0}^{n} c_{ij} T_i(u) T_j(v)$, where $a_{kl} \in \mathbb{R}^n$ ($k = 0, 1, \ldots, m$; $l = 0, 1, \ldots, n$) are the coefficients of f when written in the power basis and $c_{ij} \in \mathbb{R}^n$ ($i = 0, 1, \ldots, m$; $j = 0, 1, \ldots, n$) are the coefficients

of f when written in the Chebyshev basis. Let d_{ki}'s ($k = 0, 1, \ldots$; $i = 0, 1, \ldots, k$) be the numbers satisfying $t^k = \sum_{i=0}^{k} d_{ki} T_i(t)$. Hence,

$$\begin{aligned} f(u,v) &= \sum_{k=0}^{m}\sum_{l=0}^{n} a_{kl}\left(\sum_{i=0}^{k} d_{ki}T_i(u)\right)\left(\sum_{j=0}^{l} d_{lj}T_j(v)\right) \\ &= \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=i}^{m}\sum_{l=j}^{n} a_{kl}d_{ki}d_{lj}T_i(u)T_j(v). \end{aligned}$$

Therefore, $c_{ij} = \sum_{k=i}^{m}\sum_{l=j}^{n} a_{kl}d_{ki}d_{lj}$. This means that P_2^c can be written as

$$P_2^c = \left\{ a_{00}d_{00}d_{00} + \sum_{k=1}^{m}\sum_{l=1}^{n} a_{kl}d_{k0}d_{l0} + \sum_{i+j>0}\sum_{k=i}^{m}\sum_{l=j}^{n} a_{kl}d_{ki}d_{lj}s_{ij} : -1 \le s_{ij} \le 1 \right\},$$

But since $d_{00} = 1$,

$$\begin{aligned} P_2^c &= \left\{ a_{00} + \sum_{k=1}^{m}\sum_{l=1}^{n} a_{kl}d_{k0}d_{l0} + \sum_{j=1}^{n}\sum_{k=0}^{m}\sum_{l=j}^{n} a_{kl}d_{k0}d_{lj}s_{0j} + \right. \\ &\quad \left. \sum_{i=1}^{m}\sum_{k=i}^{m}\sum_{l=0}^{n} a_{kl}d_{ki}d_{l0}s_{i0} + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=i}^{m}\sum_{l=j}^{n} a_{kl}d_{ki}d_{lj}s_{ij} : -1 \le s_{ij} \le 1 \right\} \\ &= \left\{ a_{00} + \sum_{k=1}^{m}\sum_{l=1}^{n} a_{kl}d_{k0}d_{l0} + \sum_{k=0}^{m}\sum_{l=1}^{n}\sum_{j=1}^{l} a_{kl}d_{k0}d_{lj}s_{0j} + \right. \\ &\quad \left. \sum_{k=1}^{m}\sum_{l=0}^{n}\sum_{i=1}^{k} a_{kl}d_{ki}d_{l0}s_{i0} + \sum_{k=1}^{m}\sum_{l=1}^{n}\sum_{i=1}^{k}\sum_{j=1}^{l} a_{kl}d_{ki}d_{lj}s_{ij} : -1 \le s_{ij} \le 1 \right\} \\ &= \left\{ a_{00} + \sum_{l=1}^{n} a_{0l}\sum_{j=1}^{l} d_{lj}s_{0j} + \sum_{k=1}^{m} a_{k0}\sum_{i=1}^{k} d_{ki}s_{i0} + \right. \\ &\quad \sum_{k=1}^{m}\sum_{l=1}^{n} a_{kl}\left(d_{k0}d_{l0} + d_{k0}\sum_{j=1}^{l} d_{lj}s_{0j} + d_{l0}\sum_{i=1}^{k} d_{ki}s_{i0} + \sum_{i=1}^{k} d_{ki}\sum_{j=1}^{l} d_{lj}s_{ij} \right) : \\ &\quad \left. -1 \le s_{ij} \le 1 \right\}. \end{aligned}$$

By Lemma 3.2.1 and Lemma 3.2.2, it is seen that $-1 \le \sum_{j=1}^{l} d_{lj}s_{0j}, \sum_{i=1}^{k} d_{ki}s_{i0} \le 1$. In addition, using the fact that $|s_{ij}| \le 1$, for any $i = 0, \ldots, m$ and any $j = 0, \ldots, n$, together with Lemma 3.2.1 and Lemma 3.2.2, it is seen that $\left| d_{k0}d_{l0} + d_{k0}\sum_{j=1}^{l} d_{lj}s_{0j} + d_{l0}\sum_{i=1}^{k} d_{ki}s_{i0} + \sum_{i=1}^{k} d_{ki}\sum_{j=1}^{l} d_{lj}s_{ij} \right| \le |d_{k0}d_{l0}| +$

$|d_{k0}| \sum_{j=1}^{l} |d_{lj}| + |d_{l0}| \sum_{i=1}^{k} |d_{ki}| + \sum_{i=1}^{k} |d_{ki}| \sum_{j=1}^{l} |d_{lj}| = d_{k0}d_{l0} + d_{k0} \sum_{j=1}^{l} d_{lj} + d_{l0} \sum_{i=1}^{k} d_{ki} + \sum_{i=1}^{k} d_{ki} \sum_{j=1}^{l} d_{lj} = \left(\sum_{i=0}^{k} d_{ki}\right)\left(\sum_{j=0}^{l} d_{lj}\right) = 1$. Therefore, $P_2^c \subseteq P_2^p$. □

3.2.1 Reparametrization

The last nontrivial basis property that warrants detailed discussion is the issue of efficient reparametrization. Reparametrizing polynomials in power basis is straightforward from the *binomial* theorem. Polynomials in other bases, on the other hand, may not be as simple to reparametrize. The details of the process for polynomials in Bernstein and Chebyshev bases are covered in this section.

Reparametrization of polynomials in Bernstein basis

There is more than one algorithm to compute the reparametrization with $[0,1]^2$ of a bivariate polynomial in Bernstein basis. We describe one method here. Our method makes use of a program that, given α_{ij}'s, c, d, e, f, g, h, k, and l, computes β_{ij}'s satisfying

$$\sum_{i=0}^{m}\sum_{j=0}^{n} \alpha_{ij}(cy+d)^i(ey+f)^{m-i}(gz+h)^j(kz+l)^{n-j} = \sum_{i=0}^{m}\sum_{j=0}^{n} \beta_{ij}y^iz^j.$$

Such conversion can be done in $O\left((mn)^2\right)$ by generalizing Horner's rule. We leave the details of the conversion to the reader. Let X denote $\{(u,v) : u^0 - r \leq u \leq u^0 + r, v^0 - r \leq v \leq v^0 + r\}$. To compute the coefficients $\{\hat{c}_{ij}\}$ of $\{\hat{f}_X(\hat{u},\hat{v}) : 0 \leq \hat{u},\hat{v} \leq 1\}$, the $[0,1]^2$-reparametrized surface of $\{f(u,v) : u^0 - r \leq u \leq u^0 + r, v^0 - r \leq$

$v \le v^0 + r\}$, first substitute $u = 2r\hat{u} + u^0 - r$ and $v = 2r\hat{v} + v^0 - r$ into f, yielding

$$\begin{aligned}
f(u,v) &= \sum_{i=0}^{m}\sum_{j=0}^{n} \binom{m}{i}\binom{n}{j} c_{ij} u^i (1-u)^{m-i} v^j (1-v)^{n-j} \\
&= \sum_{i=0}^{m}\sum_{j=0}^{n} \binom{m}{i}\binom{n}{j} c_{ij} (2r\hat{u} + u^0 - r)^i \left(1 - (2r\hat{u} + u^0 - r)\right)^{m-i} \cdot \\
&\qquad (2r\hat{v} + v^0 - r)^j \left(1 - (2r\hat{v} + v^0 - r)\right)^{n-j} \\
&= \sum_{i=0}^{m}\sum_{j=0}^{n} \binom{m}{i}\binom{n}{j} c_{ij} \left((u^0 + r)\hat{u} + (u^0 - r)(1-\hat{u})\right)^i \cdot \\
&\qquad \left((1 - u^0 - r)\hat{u} + (1 - u^0 + r)(1-\hat{u})\right)^{m-i} \cdot \\
&\qquad \left((v^0 + r)\hat{v} + (v^0 - r)(1-\hat{v})\right)^j \cdot \\
&\qquad \left((1 - v^0 - r)\hat{v} + (1 - v^0 + r)(1-\hat{v})\right)^{n-j}. \qquad (3.25)
\end{aligned}$$

Substituting $\hat{u} = \tilde{u}/(\tilde{u}+1)$ and $\hat{v} = \tilde{v}/(\tilde{v}+1)$ into (3.25) yields

$$\begin{aligned}
f(u,v) &= \frac{1}{(\tilde{u}+1)^m(\tilde{v}+1)^n} \sum_{i=0}^{m}\sum_{j=0}^{n} \binom{m}{i}\binom{n}{j} c_{ij} \left((u^0 + r)\tilde{u} + u^0 - r\right)^i \cdot \\
&\qquad \left((1 - u^0 - r)\tilde{u} + 1 - u^0 + r\right)^{m-i} \cdot \\
&\qquad \left((v^0 + r)\tilde{v} + v^0 - r\right)^j \cdot \\
&\qquad \left((1 - v^0 - r)\tilde{v} + 1 - v^0 + r\right)^{n-j}. \qquad (3.26) \\
&= \frac{1}{(\tilde{u}+1)^m(\tilde{v}+1)^n} \sum_{i=0}^{m}\sum_{j=0}^{n} \gamma_{ij} \tilde{u}^i \tilde{v}^j, \qquad (3.27)
\end{aligned}$$

where (3.27) is obtained from (3.26) by the conversion program mentioned above. Substituting $\hat{u}$ and $\hat{v}$ back into (3.27) to see that

$$f(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \gamma_{ij} \hat{u}^i (1-\hat{u})^{m-i} \hat{v}^j (1-\hat{v})^{n-j}. \qquad (3.28)$$

Therefore, $\hat{c}_{ij} = \gamma_{ij}/(C(m,i)C(n,j))$ are the control points of $\hat{f}_X$ where $C(m,i) = \binom{m}{i}$.

Reparametrization of polynomials in Chebyshev basis

Let a, b, d and e be scalar constants. The reparametrization with $[-1,1]^2$ of a bivariate polynomial in Chebyshev basis can be computed if the values of λ_{ik}'s $(i = 0, 1, \ldots, m)$ satisfying

$$T_i(at+b) = \sum_{k=0}^{i} \lambda_{ik} T_k(t)$$

and the values of μ_{jk}'s $(j = 0, 1, \ldots, n)$ satisfying

$$T_j(dt+e) = \sum_{k=0}^{j} \mu_{jk} T_k(t)$$

are known. Note that

$$T_i(au+b)T_j(dv+e) = \sum_{k=0}^{i}\sum_{k'=0}^{j} \lambda_{ik}\mu_{jk'} T_k(u)T_{k'}(v),$$

which is adequate to find the reparametrization. The values of a, b, d, and e are determined by the uv-domain of the surface to be reparametrized.

To compute λ_{ik}'s, observe that for $i \geq 1$, by (3.2),

$$\begin{aligned}
T_{i+1}(at+b) &= 2(at+b)T_i(at+b) - T_{i-1}(at+b) \\
&= 2(at+b)\sum_{k=0}^{i} \lambda_{ik}T_k(t) - \sum_{k=0}^{i-1} \lambda_{i-1,k}T_k(t) \\
&= \sum_{k=0}^{i} 2a\lambda_{ik} tT_k(t) + \sum_{k=0}^{i} 2b\lambda_{ik}T_k(t) - \sum_{k=0}^{i-1} \lambda_{i-1,k}T_k(t) \\
&= 2a\lambda_{i0}tT_0(t) + \sum_{k=1}^{i} a\lambda_{ik}\left(2tT_k(t) - T_{k-1}(t)\right) + \\
&\quad \sum_{k=0}^{i-1} a\lambda_{i,k+1}T_k(t) + \sum_{k=0}^{i} 2b\lambda_{ik}T_k(t) - \sum_{k=0}^{i-1} \lambda_{i-1,k}T_k(t) \\
&= 2a\lambda_{i0}T_1(t) + \sum_{k=1}^{i} a\lambda_{ik}T_{k+1}(t) + \\
&\quad \sum_{k=0}^{i-1} a\lambda_{i,k+1}T_k(t) + \sum_{k=0}^{i} 2b\lambda_{ik}T_k(t) - \sum_{k=0}^{i-1} \lambda_{i-1,k}T_k(t), \quad (3.29)
\end{aligned}$$

and

$$T_0(at+b) = T_0(t), \tag{3.30}$$

$$T_1(at+b) = aT_1(t) + bT_0(t). \tag{3.31}$$

The equalities (3.29), (3.30), and (3.31) yield a recurrence relation of λ_{ik}'s that can be used to compute their values. The values of μ_{jk}'s can be computed similarly.

3.3 Computational results

Three versions of KTS-LS algorithms are implemented in Matlab; one operating on the polynomials in power basis, one on Bernstein basis, and one on Chebyshev basis. They are tested against a number of problem instances with varying condition numbers. Most of the test problems are created by using the normally distributed random numbers as the coefficients c_{ij}'s of f in Chebyshev basis. For some of the test problems especially those with high condition number, some coefficients are manually entered. The resulting Chebyshev polynomial system is then transformed to the equivalent system in the power and the Bernstein bases. Hence the three versions of KTS-LS solve the same polynomial system and the efficiency of the three are compared. The degrees of the test polynomials are between biquadratic and biquartic.

For the experiment, we use the algorithm by Jónsson and Vavasis [15] to compute the complex zeros required to estimate the condition number. Table 3.1 compares the efficiency of the three versions of KTS-LS for the test problems with differing condition numbers. The total number of subpatches examined by KTS-LS during the entire computation and the width of the smallest patch among those examined are reported. The results do not show any one version to be particularly

Table 3.1: Comparison of the efficiency of KTS-LS algorithm operating on the power, the Bernstein, and the Chebyshev bases. The number of patches examined during the course of the algorithm and the width of the smallest patch examined are shown for each version of KTS-LS.

cond(f)	Power basis		Bernstein basis		Chebyshev basis	
	Num. of patches	Smallest width	Num. of patches	Smallest width	Num. of patches	Smallest width
3.8×10^3	29	.125	17	.0625	21	.125
1.3×10^4	13	.125	17	.0625	13	.125
2.5×10^5	49	.0625	21	.0625	45	.0625
1.1×10^6	97	.0313	65	.0313	85	.0313
3.9×10^7	89	.0313	81	.0313	89	.0313

more efficient than the others although the Chebyshev basis has better theoretical bound than the other two.

Since the types of test polynomials may affect the relative efficiency of the three versions of KTS-LS, another experiment is performed on degree 6 univariate polynomials generated by different methods. Since Section 3.2 shows that the Chebyshev basis always gives tighter bounding polygons than the power basis, this experiment only compares between the Chebyshev and Bernstein bases. Table 3.2 and Table 3.3 show the results of this experiment. The polynomials are generated as follows. The "rand" polynomials are generated by interpolating points whose x-coordinates are evenly spaced between -1 and 1, inclusive, and whose y-coordinates are normally distributed random numbers. The "sin" ones are interpolations of $\sin(ax + b)$ at evenly spaced points between -1 and 1, inclusive, where a and b are normally distributed random numbers. The "sin-L" ones are the same as the "sin" ones except that least-squares interpolation is used instead. The "sinw" (resp. "sinw-L") ones are generated in the same way as the "sin" (resp. "sin-L") ones but with the function $\sin(6ax + b)$. Table 3.2 compares the number

Table 3.2: The numbers of test polynomials out of 1000 that bounding intervals associated with the Bernstein basis is tighter than the those associated with the Chebyshev basis, and *vice versa*.

Poly. type	Num. that Bernstein is tighter	Num. that Chebyshev is tighter
rand	1	999
sin	963	37
sin-L	960	40
sinw	436	564
sinw-L	998	2

Table 3.3: The numbers of test polynomials out of 1000 that bounding intervals associated with the Bernstein basis and those associated with the Chebyshev basis having at least one endpoint exactly at the boundary of the ranges of the polynomials.

Poly. type	Num. Bernstein with exact endpoint	Num. Chebyshev with exact endpoint
rand	2	13
sin	965	0
sin-L	972	0
sinw	330	0
sinw-L	999	658

of test polynomials of each type where one basis yields tighter bounding intervals than the other. Table 3.3 shows the number of test polynomials of each type that bounding intervals of each basis have at least one endpoint exactly at the boundary of the ranges of the polynomials. The results show that the Chebyshev basis is decidedly better for "rand" polynomials, is about the same for "sin" ones, but is worse for the rests of the polynomials than the Bernstein basis.

3.4 Summary

Three common bases, the power, the Bernstein, and the Chebyshev bases, are shown to satisfy the required properties for KTS-LS to perform efficiently. In particular, the values of θ for the three bases are derived. These values are used to calculate the time complexity of KTS-LS when that basis is used to represent the polynomial system. The Chebyshev basis has the smallest θ among the three, which shows that using KTS-LS with the Chebyshev basis has the smallest worst-case time complexity. The computational results, however, show no significant differences between the performances of the three versions of KTS-LS operating on the three bases. It appears that, in average case, choosing any of the three bases do not greatly affect the efficiency of KTS-LS. The experiment on univariate polynomials show that the Bernstein basis is more suitable for certain types of polynomials while the Chebyshev basis is better suited for other types.

Chapter 4

A Condition Number Analysis of a Surface/Surface Intersection Algorithm

We have covered the line/surface intersection problems in the previous two chapters. For this chapter, the focus now moves to the problem of finding all intersections between two parametric surfaces. The main difference in moving to surface/surface intersections is that the solutions are no longer points like in line/surface case but instead are collections of curves in most cases. This implies that algorithms for solving line/surface intersections cannot be used to solve surface/surface intersections as is.

In this chapter, we propose a new algorithm for solving surface/surface intersections. This algorithm is a modified version of KTS-LS—the algorithm for solving line/surface intersections described in Chapter 2—with certain aspects changed to handle surface/surface problems. Unlike KTS-LS, which can operate on polynomials represented in many different bases, however, the algorithm in this chapter considers only Bézier surfaces. Nevertheless, it should be able to be extended to handle other bases, too, due to its similarity to KTS-LS. The algorithm also has the same desirable feature as KTS-LS that its running time can be bounded in terms of the condition number of the problem instance.

4.1 A condition number of the surface-surface intersection problem

Let S_1 be a Bézier surface represented by

$$p(x_1, x_2) \;=\; \sum_{i=0}^{m}\sum_{j=0}^{n} a_{ij} Z_{i,m}(x_1) Z_{j,n}(x_2), \quad 0 \le x_1, x_2 \le 1,$$

where $a_{ij} \in \mathbb{R}^3$ $(i = 0, 1, \ldots, m; j = 0, 1, \ldots, n)$ denote the control points. Let S_2 be another Bézier surface represented by

$$q(x_3, x_4) \;=\; \sum_{i'=0}^{m'}\sum_{j'=0}^{n'} a'_{i'j'} Z_{i',m'}(x_3) Z_{j',n'}(x_4), \quad 0 \le x_3, x_4 \le 1,$$

where $a'_{i'j'} \in \mathbb{R}^3$. The *surface-surface intersection problem* is to find all of the intersections between the two surfaces S_1 and S_2, which are the solutions of the polynomial system

$$f(x) \equiv p(x_1, x_2) - q(x_3, x_4) \;=\; 0, \quad 0 \le x_1, x_2, x_3, x_4 \le 1,$$

where $x = (x_1, x_2, x_3, x_4)^T$. By noting that $\sum_{i=0}^{n} Z_{i,n}(t) = 1$, $f(x)$ can be written in Bernstein basis as

$$f(x) = \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{i'=0}^{m'}\sum_{j'=0}^{n'} b_{iji'j'} Z_{i,m}(x_1) Z_{j,n}(x_2) Z_{i',m'}(x_3) Z_{j',n'}(x_4), \tag{4.1}$$

where $b_{iji'j'} = a_{ij} - a'_{i'j'}$. Note that the number of control points in f is the product of the number for p and the number for q.

Following are the definitions of the quantities that are used to define the condition number of the surface-surface intersection problem. Some of these quantities are also used by our algorithm. Let $\tilde{x}$ be a point in $[0, 1]^4$ such that $f'(\tilde{x})$ has full rank. The choice of the point $\tilde{x}$ does not play a role in our algorithm but is used

in its analysis. Let $f'(\tilde{x})^\dagger = f'(\tilde{x})^T \left(f'(\tilde{x})f'(\tilde{x})^T\right)^{-1}$ be the Moore-Penrose inverse of $f'(\tilde{x})$. Note that $f'(\tilde{x})f'(\tilde{x})^\dagger = I$. Define

$$\begin{aligned}
\vartheta = \vartheta(m,n,m',n') &= \left(\sum_{i=0}^{m}\prod_{i'\neq i}\frac{\max\{|m-i'|,|i'|\}}{|i-i'|}\right)\cdot \\
&\quad \left(\sum_{j=0}^{n}\prod_{j'\neq j}\frac{\max\{|n-j'|,|j'|\}}{|j-j'|}\right)\cdot \\
&\quad \left(\sum_{i=0}^{m'}\prod_{i'\neq i}\frac{\max\{|m'-i'|,|i'|\}}{|i-i'|}\right)\cdot \\
&\quad \left(\sum_{j=0}^{n'}\prod_{j'\neq j}\frac{\max\{|n'-j'|,|j'|\}}{|j-j'|}\right) \\
&= O\left(m^{m+1}n^{n+1}(m')^{m'+1}(n')^{n'+1}\right).
\end{aligned} \tag{4.2}$$

The constant ϑ is, in fact, the equivalent of θ for polynomials with four variables. Define

$$\nu(\vartheta) = \frac{1}{16\vartheta}.$$

Let there be a Lipschitz constant $L' > 0$ for $f'(\tilde{x})^\dagger f'$ such that

$$\left\|f'(\tilde{x})^\dagger(f'(x) - f'(y))\right\| \leq L' \cdot \|x-y\| \text{ for all } x, y \in D_\vartheta,$$

where

$$D_\vartheta = [-\nu(\vartheta), 1+\nu(\vartheta)]^4.$$

The motivation of this definition of D_ϑ is that it contains all domains whose Lipschitz constants may be needed during the course of the algorithm. Let

$$L = \max\{L', 1\}.$$

Finally, define the condition number of f to be

$$\text{cond}(f) = \max_{x\in[0,1]^4} \min\left\{\frac{L+1}{\|f'(\tilde{x})^\dagger f(x)\|}, 256LN(x)\left(1+(L+1)N(x)\right)\right\}, \tag{4.3}$$

where

$$N(x) = \begin{cases} \left\| f'(x)^{\dagger} f'(\tilde{x}) \right\|, & \text{if } f'(x) \text{ has rank 3} \\ \infty, & \text{otherwise.} \end{cases}$$

Note that small condition number means the problem is well-conditioned.

The rationale for this definition of the condition number is as follows. The problem is ill-posed if there exists a point x^* on an intersection curve such that $f'(x^*)$ does not have rank 3. In that case, the surfaces intersect tangentially at x^* and slight perturbation of the surfaces can make the intersection disappears. Therefore, it is reasonable to say that the problem is ill-conditioned if there is a point x in the domain at which simultaneously $f(x)$ is close to zero and $f'(x)$ is close to having rank less than 3. If $f(x)$ is small, then the first term in the 'min' of (4.3) gets large, while if $f'(x)$ is close to rank deficient, then $N(x)$ gets large and hence so does the second term of the 'min.' In addition, $f'(\tilde{x})$ is introduced into the condition number to make it affine invariant. The condition number for the line-surface intersection problem defined in Chapter 2 is not used here because that definition seems to work only when there are a finite number of zeros.

4.2 The Kantorovich-Test Subdivision algorithm for Surface/Surface intersections

This section describes our algorithm for finding the intersections between two Bézier surfaces. Since the parametric domains of the surfaces under consideration are square, our algorithm uses the infinity norm for all of its norm computation.

During the computation, our algorithm maintains a list of *explored regions*, defined as parts of the domain $[0, 1]^4$ for which the algorithm knows for cer-

tain that they contain only the intersections that have already been found. This list is used in addition to another test to determine whether to subdivide a hypercube. We define the *Kantorovich test* on a hypercube $X = \bar{B}(x^0, r)$ as the application of Kantorovich's Theorem on the point x^0 to the function $h^{(ik)}(x) = (f(x), x_i - k)^T$ for each $i = 1, 2, 3, 4$ and any $k \in [x_i^0 - r, x_i^0 + r]$. The hypercube $\bar{B}\left(x^0, \max\{\alpha r, \nu(\vartheta)\}\right)$, $\alpha \geq 1$, is used as the domain D in the statement of the theorem, and $\left\| \left[\left(h^{(ik)}\right)'(x^0) \right]^{-1} h^{(ik)}(x^0) \right\|$ is used as η. For ω, we instead use $\hat{\omega} \geq \omega$, where $\hat{\omega}$ is defined by (4.5) below, as the minimal ω is too expensive to compute. The hypercube X passes the Kantorovich test if there exists an $i \in \{1, 2, 3, 4\}$ such that for any $k \in [x_i^0 - r, x_i^0 + r]$, $\eta\hat{\omega} \leq 1/4$ and $\bar{B}(x^0, \rho_-) \subseteq D$.

As is shown below in Section 4.4, choosing $\alpha = 1$ gives the best worst-case bound on the smallest size of subcubes our algorithm needs to examine. For average cases, however, choosing a little larger α may be better as it makes the condition $\bar{B}(x^0, \rho_-) \subseteq D$ easier to be satisfied for larger subcubes.

There are three important implications for X passing the Kantorovich test. First, x^0 is a fast starting point for $h^{(ik)}$ for the particular i that satisfies the condition of the Kantorovich test and any $k \in [x_i^0 - r, x_i^0 + r]$. Second, the segment of the intersection curve of f that corresponds to the conclusion of Kantorovich's theorem is not a loop in $x_1x_2x_3x_4$-space. Lastly, an explored region for this segment of the intersection curve can be derived. The explored region is

$$X_E = \left\{x : x_i^0 - r \leq x_i \leq x_i^0 + r\right\} \cap D \cap \bigcap_{k \in [x_i^0 - r, x_i^0 + r]} \left(\bar{B}(x^0, \rho_-^{(k)}) \cup B(x^0, \rho_+^{(k)}) \right), \tag{4.4}$$

where $\rho_-^{(k)}$ and $\rho_+^{(k)}$ are ρ_- and ρ_+ in the statement of Kantorovich's theorem with respect to $h^{(ik)}$. Observe that X_E is a hyperrectangle in $\mathbb{R}^4$ and can be stored and computed succinctly as detailed in Section 4.3.2. Note also that the explored region

provides an effective way to prevent the points on different but nearby intersection curves from being incorrectly joined into the same curve.

The other test our algorithm uses is the exclusion test. For a given hypercube X, let $\hat{f}_X$ be the Bernstein polynomial that reparametrizes with $[0,1]^4$ the surface defined by f over X. The hypercube X passes the *exclusion test* if the convex hull of the control points of $\hat{f}_X$ excludes the origin.

We now proceed to describe our algorithm, the *Kantorovich-Test Subdivision algorithm for Surface/Surface intersections* or KTS-SS in short. The algorithm maintains a queue Q of unexamined domain and a set S of explored regions throughout its computation.

Algorithm KTS-SS:

- Let Q be a queue with $[0,1]^4$ as its only entry. Set $S = \emptyset$.
- Repeat until $Q = \emptyset$
    1. Let X be the hypercube at the front of Q. Remove X from Q.
    2. If $X \not\subseteq X_{E'}$ for all $X_{E'} \in S$,
        - Perform the exclusion test on $X = \bar{B}(x^0, r)$
        - If X fails the exclusion test,
            (a) Perform the Kantorovich test on X
            (b) If X passes the Kantorovich test,
                i. Perform Newton's method on $h^{(ik)}$, where i is the index that satisfies the condition of the Kantorovich test and $k = x_i^0 - r$, starting from x^0 to find a zero x^*.

ii. Trace the segment of the intersection curve using x^* as the starting point and going toward $x_i^0 + r$ direction until the $x_i = x_i^0 + r$ boundary is reached.

iii. If the newly found segment is contained in any $X_{E'} \in S$ (i.e. the segment has been found before), discard the segment.

iv. Otherwise, compute the new explored region X_E according to (4.4). Set $S = S \cup \{X_E\}$.

(c) If either X fails the Kantorovich test or X passes the test with $X \not\subseteq X_E$, subdivide X along all four parametric axes into sixteen equal subcubes. Add these subcubes to the end of Q.

- Check if any two segments of intersection curves overlap. If so, remove the overlapping part from one of the segments.
- Join any two segments sharing an endpoint into one continuous curve. Repeat until there are no two curves sharing an endpoint.

A few remarks are needed regarding the description of the KTS-SS algorithm.

- The subdivision in step 2.c is performed in the case that X passes the Kantorovich test but $X \not\subseteq X_E$ because, in general, passing the Kantorovich test does not imply that there is only one intersection curve in X.
- The check that the intersection segment found method is not a duplicate (step 2.b.iii) is necessary since the segment detected by the Kantorovich test may be outside of X.
- By the same reason as above, certain parts of an intersection curve may be traced twice and hence must be removed from one of the segments before the segments are joined. The overlapping segments can be detected by checking

if an endpoint of a segment is inside an explored region of another segment. The segments sharing an endpoint can also be detected from explored regions in a similar manner.

- Similarly, some parts of the intersection curves found by KTS-SS may be outside $[0,1]^4$. If this behavior is undesirable, the curves that are not fully inside $[0,1]^4$ can be clipped by computing their polynomial interpolations and then solving curve/curve intersections between the interpolations and the boundaries of one of the surfaces.
- If the Kantorovich test is not applicable for a certain hypercube due to the Jacobian of the midpoint being singular, the hypercube is treated as if it fails the Kantorovich test.

Like KTS-LS, KTS-SS is also affine invariant. This is the main reason we introduce $\tilde{x}$ into the condition number: to make it affine invariant. Define $g \equiv Af$. To see that our condition number is affine invariant, note that $g'(x)^\dagger g'(y) = (Af'(x))^\dagger Af'(y) = (Af'(x))^T \left(Af'(x)\,(Af'(x))^T\right)^{-1} Af'(y) = f'(x)^T \left(f'(x)f'(x)^T\right)^{-1} f'(y) = f'(x)^\dagger f'(y)$ for any $x, y \in \mathbb{R}^4$ satisfying $f'(x)$ has full rank. Similarly, $g'(x)^\dagger g(y) = f'(x)^\dagger f(y)$ for any $x, y \in \mathbb{R}^4$ satisfying $f'(x)$ has full rank. Therefore, $\mathrm{cond}(g) = \mathrm{cond}(f)$. In contrast, simpler condition numbers such as those involving $\|f(x)\|$ and $\left\|f'(x)^\dagger\right\|$, where $x \in [0,1]^4$, are not affine invariant and hence are not chosen for our analysis.

Since Koparkar's algorithm is quite similar to KTS-SS, it is worthwhile to discuss the main differences between the two and the implications these differences make. First, Koparkar's convergence test is based on contraction mapping and evaluating ranges of functions. This test guarantees linear convergence for the simple Newton iteration. With our Kantorovich test, KTS-SS starts Newton's

method only when quadratic convergence is assured.

Another main difference is in the choice of domains for the convergence test. Koparkar's uses the subcube X itself as the domain for the test. Unless the evaluation of ranges of functions yields the *actual* ranges rather than supersets of them, this choice may exhibit undesirable behavior when a solution lies on the border of a subcube in both x_1x_2-space and x_3x_4-space at the same time, which is not necessarily on or near the border of the original domain $[0,1]^4$. In this case, any looseness of the computed bounds of ranges of functions can cause the subcube to fail the test regardless of the size of the subcube. Since no existing methods for evaluation of ranges of functions can compute the actual ranges, this results in excessive subdivisions by Koparkar's algorithm. KTS-SS uses $\bar{B}\left(x^0, \max\{\alpha r, \nu(\vartheta)\}\right)$ as the domain for X to avoid this problem. Theorem 4.4.2 below shows that the Kantorovich test does not have trouble detecting the zeros locating on the border of the subcube.

4.3 Implementation details

The implementations of certain steps of KTS-SS are not apparent and thus are explained in detail in this section. Note that only details pertaining to Bernstein basis are addressed here.

4.3.1 Computation of Lipschitz constant

For simplicity, denote $h^{(ik)}$ as h when the choice of (ik) is clear from context. Similar to line/surface case, the Lipschitz constant for $h'(x^0)^{-1}h' \equiv g$, which is

required for the Kantorovich test, is obtained from an upper bound over all $x \in X$ of the derivative of g

$$g'(x) = \left(\frac{\partial^2 \left(h'(x^0)^{-1} h \right)_i (x)}{\partial x_j \partial x_k} \right),$$

where $\left(h'(x^0)^{-1} h \right)_i (x)$ denotes the ith entry of $\left(h'(x^0)^{-1} h \right)(x)$. Let $\hat{g} \equiv \hat{g}_X$ be the Bernstein polynomial that reparametrizes with $[0,1]^4$ the surface defined by g over X. We have

$$\begin{aligned}
\max_{x \in X} \|g'(x)\| &= \max_{x \in [0,1]^4} \|\hat{g}'(x)\| \\
&= \max_{x \in [0,1]^4} \max_{\|y\|=1} \|\hat{g}'(x) y\| \\
&\leq \max_{x \in [0,1]^4} \max_{i} \sum_{j=1}^{4} \sum_{k=1}^{4} |\hat{g}'_{ijk}(x)| \\
&\leq 16 \max_{i,j,k} \max_{x \in [0,1]^4} |\hat{g}'_{ijk}(x)|.
\end{aligned}$$

Note that each entry of $\hat{g}'$ can be written as a Bernstein polynomial efficiently because

$$\frac{dZ_{i,n}(t)}{dt} = n \left(Z_{i-1,n-1}(t) - Z_{i,n-1}(t) \right),$$

where $Z_{-1,n-1}(t) = Z_{n,n-1}(t) = 0$, which can be used to compute the control points of the derivatives in Bernstein basis from a given Bernstein polynomial directly. Hence, the maximum absolute value of the control points of $\hat{g}'_{ijk}$ when written in Bernstein basis is an upper bound of $\max_{x \in [0,1]^4} |\hat{g}'_{ijk}(x)|$. Let $\hat{\omega}$ denote the Lipschitz constant computed in this manner, that is,

$$\hat{\omega} \equiv 16 \max_{i,j,k} \max_{x \in [0,1]^4} |\hat{g}'_{ijk}(x)|, \tag{4.5}$$

where $\max_{x \in [0,1]^4} |\hat{g}'_{ijk}(x)|$ is computed from its control points.

4.3.2 Implementation of the Kantorovich test and intersection curve tracing

Recall that for a hypercube X to pass the Kantorovich test, there must exist an $i \in \{1, 2, 3, 4\}$ satisfying $\eta\hat{\omega} \leq 1/4$ and $\bar{B}(x^0, \rho_-) \subseteq D$ for *all* functions $h^{(ik)}$'s where $k \in [x_i^0 - r, x_i^0 + r]$. However, the algorithm needs not explicitly check the conditions for all values of k. Notice that $\hat{\omega}$ and D are independent of k and ρ_- is an increasing function of η. For these reasons, KTS-SS only needs to check the conditions for the value of k that maximizes η. Similarly, the explored region X_E can be computed solely from the maximizer k. But note also that η is linear in k, which means that the value of k maximizing η is either $x_i^0 - r$ or $x_i^0 + r$.

After a cube passes the Kantorovich test, the segment of the intersection curve detected by the test must be traced. There are at least two acceptable methods to trace the curve segment. Since the Kantorovich test guarantees that performing Newton's method on $h^{(ik)}$ starting on x^0 converges for any $k \in [x_i^0 - r, x_i^0 + r]$, it is possible to trace the segment by repeating Newton's method starting on x^0 for many different values of k to locate the points on the segment of the intersection curve.

Another method is based on numerical integration of nonlinear ordinary differential equations describing the tangential direction of the intersection curve, which

are [24]

$$\begin{aligned}
x_1'(t) &= \frac{\det\left(c'(t), \frac{\partial p}{\partial x_2}, P(x_1, x_2)\right)}{P(x_1, x_2) \cdot P(x_1, x_2)}, \\
x_2'(t) &= \frac{\det\left(\frac{\partial p}{\partial x_1}, c'(t), P(x_1, x_2)\right)}{P(x_1, x_2) \cdot P(x_1, x_2)}, \\
x_3'(t) &= \frac{\det\left(c'(t), \frac{\partial q}{\partial x_4}, Q(x_3, x_4)\right)}{Q(x_3, x_4) \cdot Q(x_3, x_4)}, \\
x_4'(t) &= \frac{\det\left(\frac{\partial q}{\partial x_3}, c'(t), Q(x_3, x_4)\right)}{Q(x_3, x_4) \cdot Q(x_3, x_4)},
\end{aligned}$$

where

$$\begin{aligned}
P(x_1, x_2) &= \frac{\partial p}{\partial x_1} \times \frac{\partial p}{\partial x_2}, \\
Q(x_3, x_4) &= \frac{\partial q}{\partial x_3} \times \frac{\partial q}{\partial x_4}, \\
c'(t) &= \frac{P(x_1, x_2) \times Q(x_3, x_4)}{|P(x_1, x_2) \times Q(x_3, x_4)|}.
\end{aligned}$$

Using the starting point found by Newton's method on $h^{(i, x_i^0 - r)}$, the resulting initial value problem can be integrated. One point of note is that while the intersection curve may be traced to outside of X, only the segment of curve from $x_i = x_i^0 - r$ to $x_i = x_i^0 + r$ needs to be traced as the rest of the curve is traced when other hypercubes are examined.

4.3.3 Reparametrization

There are two steps of KTS-SS involving reparametrization of polynomials in Bernstein basis, namely the exclusion test and the computation of the Lipschitz constant for the Kantorovich test. Both steps require the reparametrization with $[0, 1]^4$ of Bernstein polynomials with four variables, which is a straightforward extension of reparametrization with $[0, 1]^2$ of bivariate Bernstein polynomials. See Chapter 3

for an efficient algorithm to reparametrize bivariate Bernstein polynomials with $[0,1]^2$. Alternatively, the reparametrization required by KTS-SS can be computed directly using only the reparametrization algorithm for bivariate polynomials. To reparametrize f, reparametrize p and q separately and apply (4.1) to the results. To reparametrize the Bernstein representation of $x_i - k$, reparametrize its univariate form first and compute the control points of the corresponding polynomial with four variables from it.

4.4 Time complexity analysis

In this section, we prove a number of theorems leading to the theorem regarding the running time of the KTS-SS algorithm. Since both the exclusion test and the computation of the Lipschitz constant in the Kantorovich test use the control points in their computations, it is useful to find the relationship between the control points and the function values of the polynomial defined by them. Specifically, the goal is to show that

$$\|b_{iji'j'}\| \leq \vartheta \max_{0\leq x_1,x_2,x_3,x_4\leq 1} \|f(x)\| \tag{4.6}$$

for any Bernstein polynomial f and any of its control points $b_{iji'j'} \in \mathbb{R}^d$, $d > 0$, where ϑ is as defined in (4.2). The following lemma together with Lemma (3.1.1) and Theorem (3.1.3) prove (4.6).

Lemma 4.4.1. *Let l and h be constants satisfying $l \leq h$. Assume there exists a function $\xi(m,n)$ such that*

$$\|a_{ij}\| \leq \xi(m,n) \max_{l\leq u,v\leq h} \|g(u,v)\| \tag{4.7}$$

for any a_{ij} $(i = 0,1,\ldots,m;\ j = 0,1,\ldots,n)$ and any bivariate polynomial $g(u,v) =$

$\sum_{i=0}^{n} a_{ij}\phi_i(u)\phi_j(v)$, *where* $\phi_i(u)$ *denotes the polynomial basis. Then*

$$\|b_{iji'j'}\| \le \xi(m,n)\xi(m',n') \max_{l\le u,v,s,t\le h} \|f(u,v,s,t)\| \tag{4.8}$$

for any $b_{iji'j'}$ $(i = 0,1,\ldots,m;\ j = 0,1,\ldots,n;\ i' = 0,1,\ldots,m';\ j' = 0,1,\ldots,n')$ *and any polynomial* $f(u,v,s,t) =$
$\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{i'=0}^{m'}\sum_{j'=0}^{n'} b_{iji'j'}\phi_i(u)\phi_j(v)\phi_{i'}(s)\phi_{j'}(t)$.

Proof. For any ordered pair (i',j') $(i' = 0,1,\ldots,m';\ j' = 0,1,\ldots,n')$, define a bivariate polynomial $g_{i'j'}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u)\phi_j(v)$. Let $(u^0,v^0) = \arg\max_{l\le u,v\le h}\left\|\sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u)\phi_j(v)\right\|$. By applying (4.7) to $g_{i'j'}(u,v)$,

$$\begin{aligned}\|b_{iji'j'}\| &\le \xi(m,n)\max_{l\le u,v\le h}\left\|\sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u)\phi_j(v)\right\| \\ &= \xi(m,n)\left\|\sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u^0)\phi_j(v^0)\right\|. \end{aligned}\tag{4.9}$$

Define another bivariate polynomial

$$\hat{g}(s,t) = \sum_{i'=0}^{m'}\sum_{j'=0}^{n'}\left(\sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u^0)\phi_j(v^0)\right)\phi_{i'}(s)\phi_{j'}(t).$$

By applying (4.7) to $\hat{g}(s,t)$, (4.9) becomes

$$\begin{aligned}\|b_{iji'j'}\| &\le \xi(m,n)\xi(m',n')\cdot \\ &\quad \max_{l\le s,t\le h}\left\|\sum_{i'=0}^{m'}\sum_{j'=0}^{n'}\left(\sum_{i=0}^{m}\sum_{j=0}^{n} b_{iji'j'}\phi_i(u^0)\phi_j(v^0)\right)\phi_{i'}(s)\phi_{j'}(t)\right\| \\ &\le \xi(m,n)\xi(m',n')\max_{l\le u,v,s,t\le h}\left\|\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{i'=0}^{m'}\sum_{j'=0}^{n'} b_{iji'j'}\phi_i(u)\phi_j(v)\phi_{i'}(s)\phi_{j'}(t)\right\| \\ &= \xi(m,n)\xi(m',n')\max_{l\le u,v,s,t\le h}\|f(u,v,s,t)\|. \end{aligned}$$

□

Recall that the Lipschitz constant $\hat{\omega}$ given by (4.5) is not the smallest Lipschitz constant for $h'(x^0)^{-1}h$ over $D = \bar{B}(x^0,)$. However, we can show that $\hat{\omega} \le 16\vartheta\omega$,

where ω denotes the smallest Lipschitz constant for $h'(x^0)^{-1}h$ over D. Since $\hat{\omega}$ is computed from the absolute values of the control points of $\hat{g}'_{ijk}(x)$, by (4.6),

$$\begin{aligned}\hat{\omega} &\leq 16\vartheta \max_{i,j,k} \max_{x\in[0,1]^4} \left|\hat{g}'_{ijk}(x)\right| \\ &= 16\vartheta \max_{i,j,k} \max_{x\in X} \left|g'_{ijk}(x)\right| \\ &\leq 16\vartheta \max_{x\in X} \|g'(x)\| = 16\vartheta\omega.\end{aligned}$$

With this bound on $\hat{\omega}$, we can now analyze the behavior of the Kantorovich test.

Theorem 4.4.2. *Let $f(x) = f(x_1, x_2, x_3, x_4)$ be a Bernstein polynomial system in three dimensions. Let x^0 be a point in $[0,1]^4$ such that $f'(x^0)$ has full rank. Let $r > 0$ be such that $\bar{B}(x^0, r) \subseteq [0,1]^4$. If*

$$r < \min\left\{\frac{1}{\vartheta^2 \operatorname{cond}(f)}, \frac{\nu(\vartheta)}{\alpha}\right\},$$

either

1. *The hypercube $\bar{B}(x^0, r)$ passes the Kantorovich test and the associated explored region X_E contains X, or*
2. *The hypercube $\bar{B}(x^0, r)$ passes the exclusion test.*

Proof. Let X denote the hypercube $\bar{B}(x^0, r)$. The proof is divided into two cases by the value of $(L+1)/\left(\vartheta \left\|f'(\tilde{x})^\dagger f(x^0)\right\|\right)$.

Case 1: $(L+1)/\left(\vartheta \left\|f'(\tilde{x})^\dagger f(x^0)\right\|\right) > \operatorname{cond}(f)$.

Since $\vartheta \geq 1$, $(L+1)/\left\|f'(\tilde{x})^\dagger f(x^0)\right\| > \operatorname{cond}(f)$. By definition of $\operatorname{cond}(f)$,

$$\operatorname{cond}(f) \geq 256LN(x^0)\left(1 + (L+1)N(x^0)\right). \tag{4.10}$$

By applying the assumption of this case to (4.10), we have that

$$\frac{\vartheta \left\| f'(\tilde{x})^\dagger f(x^0) \right\|}{L+1} < \frac{1}{256LN(x^0)\left(1+(L+1)N(x^0)\right)},$$

which is equivalent to

$$\begin{aligned}\frac{\vartheta \left\| f'(\tilde{x})^\dagger f(x^0) \right\|}{L+1} &< \frac{1+(L+1)N(x^0)-1}{256L(L+1)N(x^0)^2\left(1+(L+1)N(x^0)\right)} \\ &= \frac{1}{256L(L+1)N(x^0)^2} - \\ &\quad \frac{1}{256L(L+1)N(x^0)^2\left(1+(L+1)N(x^0)\right)}. \end{aligned} \tag{4.11}$$

Multiplying each term of (4.11) by $(L+1)N(x^0)$ and noting that $\left\| f'(x^0)^\dagger f(x^0) \right\| \leq \left\| f'(x^0)^\dagger f'(\tilde{x}) \right\| \cdot \left\| f'(\tilde{x})^\dagger f(x^0) \right\|$ yields

$$\vartheta \left\| f'(x^0)^\dagger f(x^0) \right\| < \frac{1}{256LN(x^0)} - \frac{1}{256LN(x^0)\left(1+(L+1)N(x^0)\right)},$$

which is equivalent to

$$64LN(x^0)\left(\vartheta \left\| f'(x^0)^\dagger f(x^0) \right\| + \frac{1}{256LN(x^0)\left(1+(L+1)N(x^0)\right)}\right) < \frac{1}{4}. \tag{4.12}$$

But $\vartheta r \leq \vartheta^2 r < 1/\operatorname{cond}(f) \leq 1/\left(256LN(x^0)\left(1+(L+1)N(x^0)\right)\right)$. Therefore, (4.12) implies

$$64\vartheta LN(x^0)\left(\left\| f'(x^0)^\dagger f(x^0) \right\| + r\right) < \frac{1}{4}. \tag{4.13}$$

Note that $LN(x^0) = L\left\| f'(x^0)^\dagger f'(\tilde{x}) \right\|$ is a Lipschitz constant of $f'(x^0)^\dagger f'$ on D_ϑ because, for any $y, z \in D_\vartheta$,

$$\begin{aligned}\left\| f'(x^0)^\dagger \left(f'(y)-f'(z)\right) \right\| &\leq \left\| f'(x^0)^\dagger f'(\tilde{x}) \right\| \cdot \left\| f'(\tilde{x})^\dagger \left(f'(y)-f'(z)\right) \right\| \\ &\leq L\left\| f'(x^0)^\dagger f'(\tilde{x}) \right\| \cdot \|y-z\|. \end{aligned}$$

Let $v(x^0)$ be the unit-length null vector of $f'(x^0)$, e.g., $f'(x^0)v(x^0) = 0$, $\|v(x^0)\| = 1$. Let i be such that $|v_i(x^0)| = 1$. Define $h(x) = (f(x), x_i - k)^T$, $x_i^0 - r \leq k \leq x_i^0 + r$.

By using the facts that $\|v(x^0)\| = 1$, $|v_i(x^0)| = 1$, and

$$h'(x^0)^{-1} = \begin{pmatrix} f'(x^0) \\ e_i^T \end{pmatrix}^{-1} = \left(\left(I - \frac{v(x^0)e_i^T}{v(x^0)^T e_i} \right) f'(x^0)^\dagger, \frac{v(x^0)}{v(x^0)^T e_i} \right),$$

where e_i denotes the ith column of the identity matrix, it is seen that

$$\eta \equiv \left\| h'(x^0)^{-1} h(x^0) \right\| \leq 2 \left(\left\| f'(x^0)^\dagger f(x^0) \right\| + r \right), \tag{4.14}$$

for any $k \in [x_i^0 - r, x_i^0 + r]$. Let ω_ϑ be the Lipschitz constant for $h'(x^0)^{-1}h'$ over D_ϑ. We have, for any $y, z \in D_\vartheta$,

$$\begin{aligned} \omega_\vartheta &= \frac{\|h'(x^0)^{-1}(h'(y) - h'(z))\|}{\|y - z\|} \\ &= \frac{1}{\|y - z\|} \cdot \left\| \left(I - \frac{v(x^0)e_i^T}{v(x^0)^T e_i} \right) f'(x^0)^\dagger (f'(y) - f'(z)) \right\| \\ &\leq \frac{2 \left\| f'(x^0)^\dagger (f'(y) - f'(z)) \right\|}{\|y - z\|} \leq 2L \left\| f'(x^0)^\dagger f'(\tilde{x}) \right\|. \end{aligned} \tag{4.15}$$

But

$$\hat{\omega} \leq 16\vartheta\omega \leq 16\vartheta\omega_\vartheta \leq 32\vartheta L \left\| f'(x^0)^\dagger f'(\tilde{x}) \right\|,$$

where ω is the Lipschitz constant for $h'(x^0)^{-1}h'$ over $D = \bar{B}(x^0, \max\{\alpha r, \nu(\vartheta)\})$ and $\hat{\omega}$ is the *computed* Lipschitz constant for $h'(x^0)^{-1}h'$ over D as defined in (4.5), because $D \subseteq D_\vartheta$. Hence, by (4.13),

$$\eta\hat{\omega} < \frac{1}{4}, \tag{4.16}$$

for any $k \in [x_i^0 - r, x_i^0 + r]$, which is one of the conditions for X to pass the Kantorovich test.

For the other condition, note that $\sqrt{1 - 2h} \geq 1 - 2h$ for $0 \leq h \leq 1/2$. Therefore,

$$\begin{aligned} \rho_- &= \frac{1 - \sqrt{1 - 2\eta\hat{\omega}}}{\hat{\omega}} \\ &\leq 2\eta \\ &\leq \frac{1}{64\vartheta L N(x^0)} \end{aligned} \tag{4.17}$$

by (4.13) and (4.14). Also note that

$$\left\|f'(x^0)^\dagger f'(x^0)\right\| \le \left\|f'(x^0)^\dagger f'(\tilde{x})\right\| \cdot \left\|f'(\tilde{x})^\dagger f'(x^0)\right\| = N(x^0)\left\|f'(\tilde{x})^\dagger f'(x^0)\right\|.$$

Since $f'(x^0)^\dagger f'(x^0)$ is symmetric (by definition of Moore-Penrose inverse) and $\left(f'(x^0)^\dagger f'(x^0)\right)^2 = f'(x^0)^\dagger f'(x^0)$, $f'(x^0)^\dagger f'(x^0)$ is a (nonzero) orthogonal projection matrix. This means $\left\|f'(x^0)^\dagger f'(x^0)\right\|_2 = 1$. Therefore, (4.17) becomes

$$\begin{aligned}
\rho_- &\le \frac{\left\|f'(\tilde{x})^\dagger f'(x^0)\right\|}{64\vartheta L\left\|f'(x^0)^\dagger f'(x^0)\right\|} \\
&\le \frac{\left\|f'(\tilde{x})^\dagger f'(x^0)\right\|}{32\vartheta L\left\|f'(x^0)^\dagger f'(x^0)\right\|_2} \\
&\le \frac{\left\|f'(\tilde{x})^\dagger f'(x^0)\right\|}{32\vartheta L}.
\end{aligned}$$

But for any $y \in [0,1]^4$,

$$\begin{aligned}
\left\|f'(\tilde{x})^\dagger f'(y)\right\| &= \left\|f'(\tilde{x})^\dagger\left(f'(y) - f'(\tilde{x})\right) + f'(\tilde{x})^\dagger f'(\tilde{x})\right\| \\
&\le L\left\|y - \tilde{x}\right\| + 1 \\
&\le L + 1. \qquad (4.18)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\rho_- &\le \frac{L+1}{32\vartheta L} \\
&\le \frac{1}{16\vartheta} = \nu(\vartheta),
\end{aligned}$$

showing that X passes the Kantorovich test.

Finally, the associated explored region X_E contains X because

$$\begin{aligned}
\rho_+ &= \frac{1+\sqrt{1-2\eta\hat{\omega}}}{\hat{\omega}} \\
&\geq \frac{1}{\hat{\omega}} \\
&\geq \frac{1}{32\vartheta LN(x^0)} \\
&\geq \frac{1}{256\vartheta LN(x^0)\left(1+(L+1)N(x^0)\right)} \\
&\geq \frac{1}{\vartheta\,\mathrm{cond}(f)} \\
&> \vartheta r \geq r,
\end{aligned}$$

for any $k \in [x_i^0 - r, x_i^0 + r]$.

Case 2: $(L+1)/\left(\vartheta \left\|f'(\tilde{x})^\dagger f(x^0)\right\|\right) \leq \mathrm{cond}(f)$.

By (4.18) and the assumption of this case, it is seen that

$$\begin{aligned}
r &< \frac{1}{\vartheta^2\,\mathrm{cond}(f)} \\
&\leq \frac{\left\|f'(\tilde{x})^\dagger f(x^0)\right\|}{\vartheta(L+1)} \\
&\leq \frac{\left\|f'(\tilde{x})^\dagger f(x^0)\right\|}{\vartheta \max_{y\in X}\left\|f'(\tilde{x})^\dagger f'(y)\right\|},
\end{aligned}$$

which is equivalent to

$$\vartheta \cdot \max_{y\in X}\left\|f'(\tilde{x})^\dagger f'(y)\right\| \cdot r < \left\|f'(\tilde{x})^\dagger f(x^0)\right\|. \tag{4.19}$$

Recall that $\max_{y\in X}\left\|f'(\tilde{x})^\dagger f'(y)\right\|$ is the Lipschitz constant for $f'(\tilde{x})^\dagger f$ on X. Hence, for any $x \in X$,

$$\begin{aligned}
\left\|f'(\tilde{x})^\dagger f(x) - f'(\tilde{x})^\dagger f(x^0)\right\| &\leq \max_{y\in X}\left\|f'(\tilde{x})^\dagger f'(y)\right\| \cdot \left\|x - x^0\right\| \\
&\leq \max_{y\in X}\left\|f'(\tilde{x})^\dagger f'(y)\right\| \cdot r.
\end{aligned} \tag{4.20}$$

Combining (4.19) and (4.20) gives

$$\vartheta \cdot \left\| f'(\tilde{x})^\dagger f(x) - f'(\tilde{x})^\dagger f(x^0) \right\| < \left\| f'(\tilde{x})^\dagger f(x^0) \right\|. \tag{4.21}$$

Define $\hat{f}(\hat{x})$ such that

$$\begin{aligned} \hat{f}(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4) &= f(2r\hat{x}_1 + x_1^0 - r, 2r\hat{x}_2 + x_2^0 - r, \\ &\qquad 2r\hat{x}_3 + x_3^0 - r, 2r\hat{x}_4 + x_4^0 - r). \end{aligned} \tag{4.22}$$

In other word, $\hat{f}$ is a Bernstein polynomial that reparametrizes with $[0,1]^4$ the surface defined by f over X. In terms of $\hat{f}$, (4.21) is equivalent to

$$\vartheta \cdot \left\| f'(\tilde{x})^\dagger \hat{f}(\hat{x}) - f'(\tilde{x})^\dagger \hat{f}(\hat{x}^0) \right\| < \left\| f'(\tilde{x})^\dagger \hat{f}(\hat{x}^0) \right\|$$

for some $\hat{x} \in [0,1]^4$, where $\hat{x}$ is the rescaled x and $\hat{x}^0$ is the rescaled x^0 according to (4.22). In particular,

$$\vartheta \cdot \max_{\hat{x} \in [0,1]^4} \left\| f'(\tilde{x})^\dagger \hat{f}(\hat{x}) - f'(\tilde{x})^\dagger \hat{f}(\hat{x}^0) \right\| < \left\| f'(\tilde{x})^\dagger \hat{f}(\hat{x}^0) \right\|. \tag{4.23}$$

Let $l(\hat{x}) \equiv f'(\tilde{x})^\dagger \hat{f}(\hat{x})$ and $g(\hat{x}) \equiv l(\hat{x}) - l(\hat{x}^0)$. By (4.6),

$$\|c_{iji'j'}\| \leq \vartheta \cdot \max_{\hat{x} \in [0,1]^4} \|g(\hat{x})\|,$$

for any control point $c_{iji'j'}$ of $g(\hat{x})$, which is equivalent to

$$\left\| a_{iji'j'} - l(\hat{x}^0) \right\| \leq \vartheta \cdot \max_{\hat{x} \in [0,1]^4} \left\| l(\hat{x}) - l(\hat{x}^0)) \right\|, \tag{4.24}$$

for any control point $a_{iji'j'}$ of $l(\hat{x})$. Substituting (4.24) into the left hand side of (4.23) yields

$$\left\| a_{iji'j'} - l(\hat{x}^0) \right\| < \left\| l(\hat{x}^0) \right\|,$$

which implies that the convex hull of the control points of $l(\hat{x})$ does not contain the origin. Since $f'(\tilde{x})l(x) = f'(\tilde{x})f'(\tilde{x})^\dagger \hat{f}(\hat{x}) = \hat{f}(\hat{x})$ and the convex hull of the control points of a Bernstein polynomial is affinely invariant (See Chapter 3), the convex hull of the control points of $\hat{f}(\hat{x})$ does not contain the origin, either. Therefore, X passes the exclusion test. □

One remark regarding Theorem 4.4.2 is that, in most cases, $1/\left(\vartheta^2 \operatorname{cond}(f)\right) \leq 1/(16\vartheta\alpha) = \nu(\vartheta)/\alpha$. Only when $\operatorname{cond}(f)$ is very small (i.e., f is highly well-conditioned) or ϑ is small (i.e., the surfaces are planes) that $1/\left(\vartheta^2 \operatorname{cond}(f)\right)$ may be larger than $\nu(\vartheta)/\alpha$.

4.5 Computational results

The KTS-SS algorithm is implemented in Matlab and is tested against a number of problem instances with varying condition numbers. We estimate the condition number by computing L using the method in Section 4.3.1 and uniformly sampling points from $[0,1]^4$ to compute (4.3). The point $\tilde{x}$ is chosen to be $(.25, .75, .5, .5)^T$ as it is a common valid point among all of the test cases. From our experiments, the choice of $\tilde{x}$ does not significantly affect the condition number as it never changes the condition number by more than a factor of 10.

Table 4.1 compares the efficiency of KTS-SS for each test problem with its condition number. The total number of subcubes examined by KTS-SS during the entire computation, the width of the smallest hypercube among those examined, and the maximum number of Newton iterations to converge are reported. Note that the high number of Newton iterations of some test cases is due to roundoff error. Some test problems and their solutions are also shown in Figure 4.1–4.5.

4.6 Summary

We present KTS-SS algorithm for finding the intersections between two Bézier surfaces. By using the combination of subdivision and Kantorovich's theorem, our

Table 4.1: Efficiency of KTS-SS algorithm on problems of different condition numbers.

Fig.	$\max\{m, n, m', n'\}$	cond(f)	Num. cubes	Smallest width	Max. Newton iter.
-	2	3.06×10^3	449	.06250	3
-	2	7.00×10^3	497	.03125	3
4.1	2	1.89×10^4	2641	.00097	4
-	2	2.55×10^5	1585	.00391	6
4.2	3	1.81×10^4	145	.06250	-
4.3	3	5.10×10^4	1729	.00781	4
-	3	7.81×10^6	145	.03125	4
-	3	2.20×10^7	5105	.00391	6
-	3	1.83×10^9	7425	.00391	10
4.4	3	3.15×10^9	2609	.00781	9
4.5	4	3.01×10^{10}	4241	7.63×10^{-6}	4

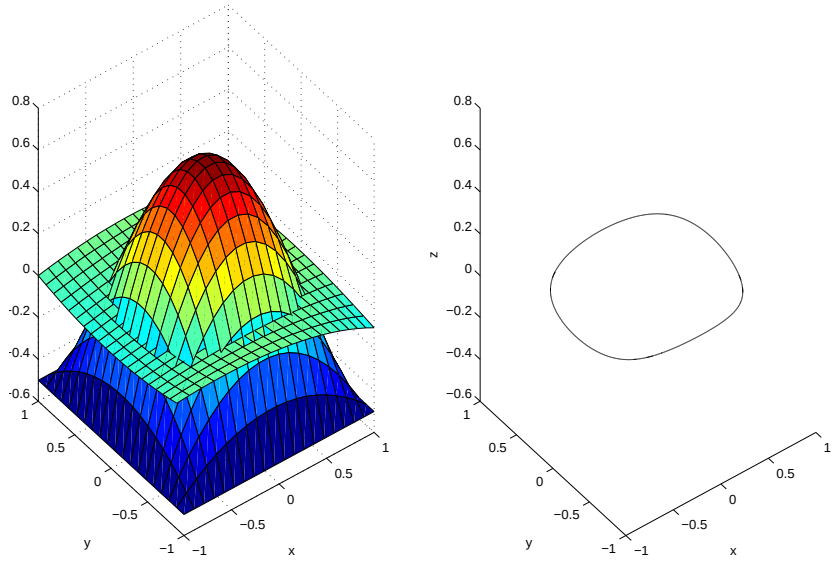


Figure 4.1: Surfaces of test case 3 and their intersections.

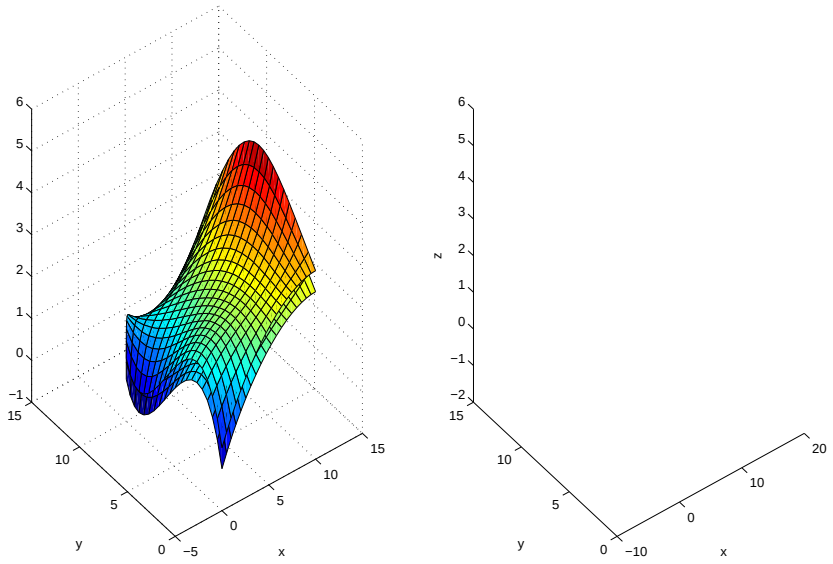


Figure 4.2: Surfaces of test case 5, which do not intersect.

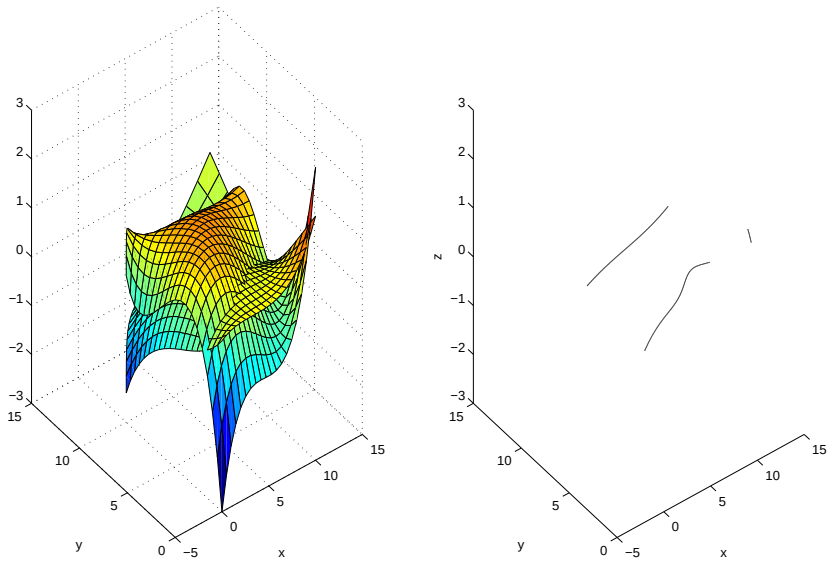


Figure 4.3: Surfaces of test case 6 and their intersections.

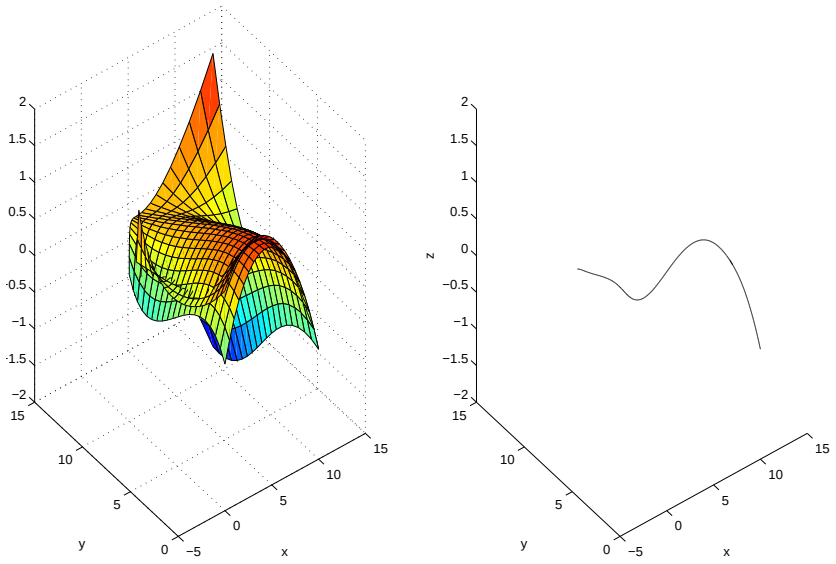


Figure 4.4: Surfaces of test case 10 and their intersections.

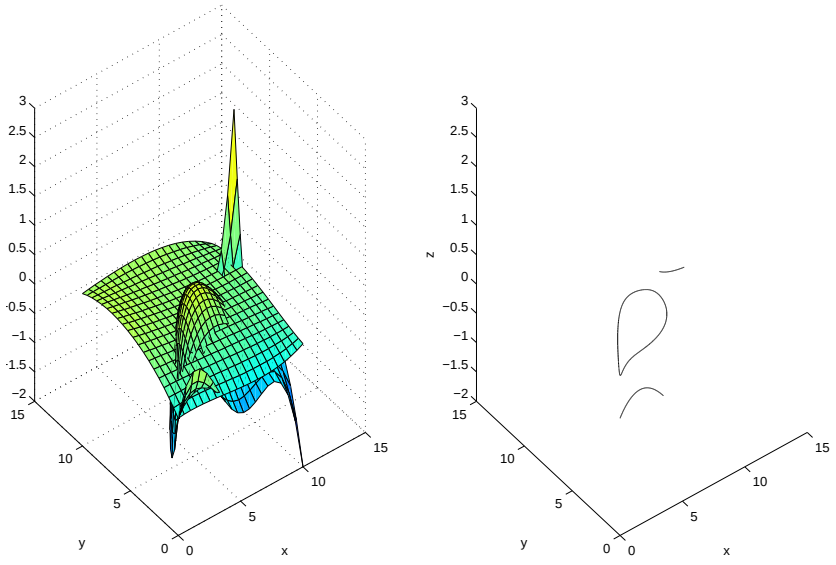


Figure 4.5: Surfaces of test case 11 and their intersections.

algorithm can take advantage of the quadratic convergence of Newton's method without the problems of divergence and missing some intersections that commonly occur with Newton's method. KTS-SS is very similar to KTS-LS. The main difference is in the Kantorovich test and the additional step to merge intersection segments together. Finally, we show that the efficiency of KTS-SS has an upper bound that depends solely on the condition number of the problem.

Chapter 5

Discussion

New algorithms for solving line/surface and surface/surface intersections are proposed. Both of them are hybrid of subdivision methods and Newton's method, using tests based on Kantorovich's theorem to assure quadratic convergence of Newton iterations. These algorithms enjoy the desirable property that their running time has an upper bound in terms of the condition number of the problem instance and the constant depending on the basis representing the polynomials. To our knowledge, there are no other algorithms in literature that are shown to have this property. Furthermore, some algorithms are easily seen to not have this property.

Our line/surface algorithm KTS-LS can operate on polynomials represented in any bases with certain properties. The power, Bernstein, and first-kind Chebyshev bases are examples of those suitable for the algorithm. Using the Chebyshev basis with KTS-LS is shown to have the best theoretical bound on its running time among the three. However, the computational results do not show significant difference in running time between using Chebyshev basis and using the others. More investigation is needed to explain this phenomenon.

A number of questions are left unanswered by this dissertation, however, such as

- **Extensibility to piecewise polynomial surfaces and/or NURBS**. Since KTS-LS and KTS-SS only require the ability to compute the bounding polytope of a subdomain that satisfies the list of basis properties, it may

be possible to extend the algorithms to handle these more general surfaces if bounding polytopes having similar properties can be computed relatively quickly.

- **Tighter condition numbers**. The condition numbers presented earlier seem loose. It is likely that tighter condition numbers exist. If tighter condition numbers are found, we would be able to calculate tighter bounds on the time complexity of both algorithms, too.
- **Using KTS-LS and KTS-SS in floating point arithmetic**. In the presence of roundoff error, we may need to make adjustments for both algorithms to be able to guarantee that all intersections are found. In addition, the accuracy of the computed intersections would become an important issue in this case.
- **Using expression trees with KTS-SS**. Elber and Grandine propose the use of expression trees to reduce the number of constraints of the problem, for example, the number of coefficients needed to represent polynomials, in many applications [6]. Can KTS-SS operate on polynomials represented in the form of expression tree constraints to take advantage of reduced number of coefficients?
- **The necessity of the generic coefficients assumption in the analysis of KTS-LS**. Is it possible to analyze the efficiency of KTS-LS without this assumption?
- **Choice of polynomial basis**. As mentioned previously, Chebyshev basis has the best (smallest) value of θ, and therefore ought to require the fewest number of subdivisions. The computational results in Chapter 3, however, do not indicate a clear-cut advantage for the Chebyshev basis. Our conjecture is that the problem instances where Bernstein case performs very poorly are

few, and for other instances, the Bernstein case performs just as well as the Chebyshev. Nevertheless, further research is needed to correctly explain this phenomenon and understand the impact of the choice of basis on practical efficiency of KTS-LS.

- **Handling singularities and degeneracy**. Due to the algorithms' reliance on Newton's method, they cannot handle singularities and degeneracy in the polynomials. When the algorithms encounter a singular point, the algorithms do not terminate as the convergence tests can never be satisfied. Similar situation happens in degenerate cases. Yap proposes a subdivision-type algorithm for solving Bézier curves intersection problems that correctly handles all degenerate cases [38]. The interesting question is how to handle singularities and degeneracy in KTS-LS and KTS-SS without sacrificing the property that their running time is bounded only in terms of the condition numbers.

BIBLIOGRAPHY

[1] C. L. Bajaj, C. M. Hoffmann, J. E. Hopcroft, and R. E. Lynch. Tracing surface intersections. *Computer Aided Geometric Design*, 5(4):285–307, 1988.

[2] R. E. Barnhill and S. N. Kersey. A marching method for parametric surface / surface intersection. *Computer Aided Geometric Design*, 7(1–4):257–280, 1990.

[3] W. Barth and W. Stürzlinger. Efficient ray-tracing for Bézier and B-spline surfaces. *Computers and Graphics*, 17(4):423–430, 1993.

[4] K. P. Cheng. Using plane vector fields to obtain all the intersection curves of two general surfaces. In *Theory and practice of Geometric Modeling*, pages 187–204. Springer-Verlag, New York, 1989.

[5] Peter Deuflhard and Gerhard Heindl. Affine invariant convergence theorems for Newton's method and extensions to related methods. *SIAM J. Numer. Anal.*, 16:1–10, 1980.

[6] G. Elber and T. Grandine. Efficient solution to systems of multivariate polynomials using expression trees. Presentation at the 10th SIAM Conference on Geometric Design and Computing, San Antonio, TX., 2007.

[7] I. Z. Emiris and J. F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symb. Comput.*, 20:117–149, 1995.

[8] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 5 edition, 2002.

[9] Rida T. Farouki, Bethany K. Kuspa, Carla Manni, and Alessandra Sestini. Efficient solution of the complex quadratic tridiagonal system for $\mathcal{C}^2$ PH quintic splines. *Numerical Algorithms*, 27:35–60, 2001.

[10] Alain Fournier and John Buchanan. Chebyshev polynomials for boxing and intersections of parametric curves and surfaces. *Computer Graphics Forum*, 13(3):C–127–C–142, 1994.

[11] Robert M. Freund and Jorge R. Vera. Condition-based complexity of convex optimization in conic linear form via the ellipsoid algorithm. *SIAM J. Optim.*, 10:155–176, 1999.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. the Johns Hopkins University Press, 3 edition, 1996.

[13] T. A. Grandine and F. W. Klein. A new approach to the surface intersection problem. *Computer Aided Geometric Design*, 14(2):111–134, 1997.

[14] E. G. Houghton, R. F. Emnett, J. D. Factor, and C. L. Sabharwal. Implementation of a divide-and-conquer method for intersection of parametric surfaces. *Computer Aided Geometric Design*, 2(1–3):173–183, 1985.

[15] G. Jónsson and S. Vavasis. Accurate solution of polynomial equations using Macaulay resultant matrices. *Mathematics of Computation*, 74:221–262, 2005.

[16] James T. Kajiya. Ray tracing parametric patches. *SIGGRAPH Comput. Graph.*, 16(3):245–254, 1982.

[17] L. Kantorovich. On Newton's method for functional equations (Russian). *Dokl. Akad. Nauk SSSR*, 59:1237–1240, 1948.

[18] P. Koparkar. Surface intersection by switching from recursive subdivision to iterative refinement. *The Visual Computer*, 8:47–63, 1991.

[19] G. A. Kriezis, N. M. Patrikalakis, and F.-E. Wolter. Topological and differential-equation methods for surface intersections. *Computer-Aided Design*, 24(1):41–55, 1992.

[20] Daniel Lischinski and Jakob Gonczarowski. Improved techniques for ray tracing parametric surfaces. *The Visual Computer*, 6:134–152, 1990.

[21] Y. Ma and Y.-S. Lee. Detection of loops and singularities of surface intersections. *Computer-Aided Design*, 30(14):1059–1067, 1998.

[22] T. Nishita, T. W. Sederberg, and M. Kakimoto. Ray tracing trimmed rational surface patches. *ACM Computer Graphics*, 24(4):337–345, 1990.

[23] N. M. Patrikalakis. Surface-to-surface intersections. *IEEE Computer Graphics and Applications*, 13(1):89–95, 1993.

[24] N. M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag Berlin Heidelberg, Germany, 2002.

[25] K. H. Qin, M. L. Gong, Y. J. Guan, and W. P. Wang. A new method for speeding up ray tracing NURBS surfaces. *Computers and Graphics*, 21(5):577–586, 1997.

[26] J. R. Rossignac and A. A. G. Requicha. Piecewise-circular curves for geometric modeling. *IBM Journal of Research and Development*, 31(3):296–313, 1987.

[27] Steven M. Rubin and Turner Whitted. A 3-dimensional representation for fast rendering of complex scenes. *SIGGRAPH Comput. Graph.*, 14(3):110–116, 1980.

[28] T. W. Sederberg, H. N. Christiansen, and S. Katz. Improved test for closed loops in surface intersections. *Computer-Aided Design*, 21(8):505–508, 1989.

[29] E. C. Sherbrooke and N. M. Patrikalakis. Computation of the solutions of nonlinear polynomial systems. *Computer Aided Geometric Design*, 10(5):379–405, 1993.

[30] Gun Srijuntongsiri and Stephen A. Vavasis. A condition number analysis of a line-surface intersection algorithm. *SIAM Journal on Scientific Computing*, 2008. To appear.

[31] M. A. J. Sweeney and R. H. Bartels. Ray tracing free-form B-spline surface. *IEEE Computer Graphics and Application*, 6:41–49, 1986.

[32] Kim-Chuan Toh and Lloyd N. Trefethen. Calculation of pseudospectra by the Arnoldi iteration. *SIAM J. Sci. Comput.*, 17:1–15, 1996.

[33] Daniel L. Toth. On ray tracing parametric surfaces. *SIGGRAPH Comput. Graph.*, 19(3):171–179, 1985.

[34] S. W. Wang, Z. C. Shih, and R. C. Chang. An efficient and stable ray tracing algorithm for parametric surfaces. *Journals of Information Science and Engineering*, 18(4):541–561, 2002.

[35] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

[36] S.-T. Wu and L. N. Andrade. Marching along a regular surface/surface intersection with circular steps. *Computer Aided Geometric Design*, 16(4):249–268, 1999.

[37] C. G. Yang. On speeding up ray tracing of B-spline surfaces. *Computer Aided Design*, 19:122–130, 1987.

[38] Chee K. Yap. Complete subdivision algorithms, i: intersection of bezier curves. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 217–226, New York, NY, USA, 2006. ACM.